(Nothing else) MATor(s):
Monitoring the Anonymity of Tor's Path Selection

Michael Backes[1,2] Aniket Kate[1,3]

Sebastian Meiser[1,2] Esfandiar Mohammadi[1,2]

[1] Saarland University, [2] CISPA, [3] MMCI
{backes, meiser, mohammadi}@cs.uni-saarland.de
aniket@mmci.uni-saarland.de

October 14, 2014

Abstract

In this paper we present MATOR: a framework for rigorously assessing the degree of anonymity in the Tor network. The framework explicitly addresses how user anonymity is impacted by real-life characteristics of actually deployed Tor, such as its path selection algorithm, Tor consensus data, and the preferences and the connections of the user. The anonymity assessment is based on rigorous anonymity bounds that are derived in an extension of the ANOA framework (IEEE CSF 2013). We show how to apply MATOR on Tor's publicly available consensus and server descriptor data, thereby realizing the first real-time anonymity monitor. Based on experimental evaluations of this anonymity monitor on Tor Metrics data, we propose an alternative path selection algorithm that provides stronger anonymity guarantees without decreasing the overall performance of the Tor network.

Contents

1 Introduction

The onion routing network Tor is a widely employed low-latency anonymous communication service [30]. To provide anonymity Tor routes a user's traffic through anonymizing proxies. In Tor the trust in these anonymizing proxies (also called nodes) is distributed over three nodes, instead of one proxy, which are chosen from more than 5000 volunteer nodes. Using these anonymizing proxies, Tor creates an anonymous channel for the user, which leads to the following central question from a user perspective:

> How anonymous is this channel that Tor creates, i.e., how likely is it that an adversary can deanonymize me?

Deriving the degree of a user's anonymity is challenging for such a complex system where each of the 5000 fluctuating nodes is entrusted with different bandwidth, and each node offers a different set of ports for a communication. Previous mathematically founded analyses abstract the Tor network by ignoring characteristics of Tor, such as the path selection algorithm, the varying entrusted bandwidth of different Tor nodes, or the user's requested ports [6,13–16,19]. However, these real-life characteristics of Tor significantly influence a user's anonymity, which renders the previously proven bounds inaccurate.

Contribution. In this paper, we present MATor: the first system to derive sender, recipient and relationship anonymity guarantees based on Tor's real-life characteristics, such as its actual path selection strategy. Our anonymity definitions are founded in the AnoA framework [6] and are thus modular. MATor entails light-weight real-time monitors that compute sender, recipient and relationship anonymity guarantees based on the actual Tor consensus data and the user requested ports.

As AnoA, the theoretical framework upon which we base MATor, does not provide a formalism to model adversaries that resemble the worst case for a realistic scenario, we extend AnoA with the concepts of adversary classes and adaptive user behavior. These adversary classes allow for restricting the strong AnoA adversary to the scenario of interest, whereas extending AnoA with adaptive user behavior enables interactive communication scenarios. We show that sequential composition does not hold for some adversary classes and characterize those adversary classes for which we show sequential composition. We consider this extension of the AnoA framework to be of independent interest for the analysis of other anonymous communication networks.

Using MATor, we conduct experiments on Tor Metrics [29] data that span the last 24 months. These experiments illustrate that the anonymity guarantees fluctuate substantially on an hourly basis, which further underscores the value of MATor's novel methodology to monitor anonymity in real-time. The experiments additionally highlight the influence of the entrusted bandwidth on a user's anonymity, as Tor's path selection algorithm places a large amount of trust into single high-bandwidth nodes and hence has similar weaknesses as choosing a single anonymizing proxy. Based on our findings from the experiments, we propose DistribuTor: a novel path selection algorithm that improves Tor for all three considered anonymity notions (sender, recipient and relationship anonymity) by redistributing the trust in single high-bandwidth nodes to a larger set of Tor nodes. In contrast to previous proposals for path selection algorithms that solely concentrate on improving the users' anonymity [4,11], our path selection algorithm additionally preserves the overall performance of the Tor network.

Related work. In the literature, there are two lines of work about estimating the degree of anonymity of Tor users.

The first line of work assumes a worst case adversary and proves rigorous anonymity bounds for Tor users [6, 13–16, 19]. These pieces of work, however, abstract the Tor network by ignoring characteristics of Tor that significantly influence a user's anonymity, such as the path selection algorithm, the varying entrusted bandwidth of different Tor nodes, or the user's requested ports. Nevertheless, these previous approaches for proving anonymity degrees for Tor offer a valuable foundation for our work: in particular, the rigorous AnoA framework. AnoA, however, has several severe limitations: it only considers a worst-case adversary that has full control over all actions of all users, which is too strong for many realistic and interesting scenarios. However, there are important scenarios in which the adversary should not have full control over all actions of all users and that cannot be modeled in AnoA. Therefore, we extend AnoA such that it also allows much more fine-grained scenarios.

The second line of work models Tor's anonymity relevant characteristics more accurately [22, 31], e.g., they explicitly model Tor's path selection. Johnson et al. [22] give estimates for the anonymity that Tor provides for typical users based on simulation experiments that assume an experimentally estimated adversarial strategy for compromising relays, which, however, does not model the worst case adversary. Moreover, they characterize the adversary by its bandwidth and split this bandwidth up between entry and exit nodes (i.e., the first and third proxy), which neglects the following case: a relay that offers its bandwidth as entry node will with significant probability be chosen as a middle node (i.e., the second proxy) or as an exit node (i.e., the third proxy) and can still learn some information. This case is ignored by their analysis, which significantly reduces the estimated de-anonymization probability of adversarial relays. The work by Wacek et al. [31] goes into a similar direction. They construct

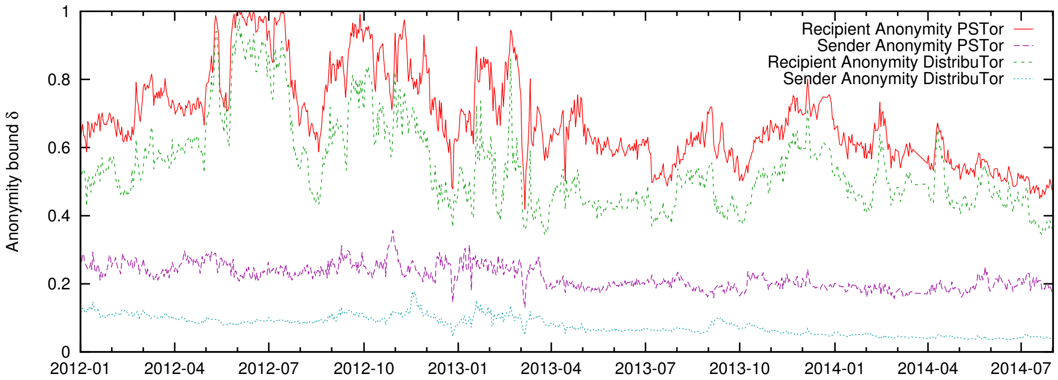

Figure 1: Output of MATor on historical data from January 2012 until the end of July 2014: HTTPS + IRC vs. HTTPS, The graph shows a bound on the success probability of a worst case adversary in deanonymizing a user, where the adversary may compromise up to 0.5% of the nodes.

a small-scale model of the Tor network, on which they perform extensive simulations, using the real Tor client code, to evaluate the impact of path selection algorithms on Tor's performance and anonymity. As both analyses are based on simulations, they, in contrast to our work, do not yield provable bounds on the degree of a user's anonymity. Moreover, unlike MATor, their analyses cannot be performed in real time on commodity hardware.

With respect to the improvement on Tor's path selection algorithm, there is an extensive line of work. Wang et al. [32] focus on improving Tor's latency by using a congestion-aware path selection. This proposed path selection would improve the overall performance of Tor, but the authors show that their path selection does not improve a user's anonymity and even slightly reduces it. In contrast, there are several proposals to improve the anonymity of single user's in Tor significantly [4,11,23], but these path selection algorithms solely aim to improve the anonymity of Tor and ignore the influence on the overall performance of the Tor network. If one of these latter path selection algorithms was to be widely deployed and used by the majority of Tor users, the overall performance of the Tor network would significantly decrease, as the path selection would no longer allow the traffic to be distributed according to the bandwidth of the nodes. In this work, we propose a path selection algorithm that both significantly improves the anonymity of users and at the same time is completely performance-preserving; i.e., even if every Tor user applied our technique, the overall performance of Tor would not change at all.

2 Overview

The Tor network is an overlay network consisting of OR *nodes* that relay user traffic and of a distributed directory service that maintains and provides the users with cryptographic information and with routing information about the OR nodes in form of a consensus. On a regular basis (typically every 10 minutes) users select a sequence of (three) OR nodes based on the directory information, such as bandwidth and accepted ports, and create a path, called a *circuit* over the selected set. The users then employ these circuits to send traffic to their destinations and to receive responses from those. The selection of nodes is done by a random choice. However, to improve the performance of the Tor network, this choice is not uniformly at random, but a weighted random choice over all possible nodes, where the weight depends on the node's bandwidth.

Our goal is to perform a rigorous real-time analysis of a user's anonymity bounds in the presence of real-life parameters such as the current path selection algorithm in a mathematically founded framework.

Due to the lack of a suitable theoretical framework that we could use off-the-shelf, we chose to extend the AnoA framework [6], a general framework for anonymity guarantees, such that we can prove our analysis secure. AnoA is a mathematical framework that is conceptually based on differential privacy in three ways: first, it considers the user behavior as a database of user actions, hence statically determines a users' behaviors; second, it leaves the choice over a user's behavior to the adversary; and third, it allows for deriving guarantees for all sufficiently similar user behaviors. By statically determining the behavior of all users (in a database) before this behavior is executed, AnoA excludes interactive communication scenarios.

In Section 3 we extend the AnoA framework to such interactive communication scenarios by allowing the adversary to specify user actions as atomic steps instead of providing databases. In addition, we restrict the adversary's power in AnoA by introducing wrapper machines that we coin *adversary classes* for including settings with uncompromised servers. With these adversary classes we, additionally, are able to describe usage patterns and to restrict the ports that an adversary may choose for a user's connection and the influence that the adversary may have on user preferences. With adversary classes, a variety of user profiles and realistic restrictions for the adversary can be defined. In addition, they can modularly be combined with anonymity notions, such as sender anonymity, recipient anonymity and relationship anonymity.

Before we discuss how a user's anonymity in Tor is assessed in MATOR, we describe in Section 4 Tor and its path selection in order to illustrate that there is a variety of parameters that need to be carefully considered. Then, we finally define, on top of ANOA, MATOR as a framework for assessing a user's anonymity in the Tor network. We describe how to derive bounds on the degree of anonymity, depending on the path selection algorithm and actual Tor consensus data. From this data we derive the impact of individual relays on anonymity – which is the likelihood that they are used as entry, middle or exit node in a user's circuit – depending on the user's preferences, the ports she wants to connect to and the structure of the Tor network, such as the active Tor nodes with their tags, entrusted bandwidths and family relations [29]. We then base the computation of bounds for sender, recipient, and relationship anonymity on these calculations, and we show these bounds to be founded in ANOA. To allow users to compute their anonymity, we present light-weight live-monitors that implement MATOR for sender, recipient and relationship anonymity.

In Section 6 we conduct experiments by applying MATOR to Tor network data, taken from the Tor Metrics archive [29]. These experiments give several interesting insights into Tor's anonymity guarantees: these guarantees heavily depend on a user's preferences, Tor's path selection algorithm and the status of all active Tor nodes. Consequently, the guarantees vary over time as new nodes become active, are entrusted with more or less bandwidth and receive different tags. As depicted in Figure 12, these guarantees significantly fluctuate with every new consensus file, i.e., every hour. Moreover, our experiments show that the current path selection algorithm does not properly distribute the trust over Tor nodes and extremely prioritizes a few high-bandwidth nodes. This leads to single points of failure and thus decreases the anonymity guarantees we can derive. Based on this insight, we propose an alternative path selection algorithm, coined DISTRIBUTOR, that maximally distributes the trust without decreasing the overall performance of the Tor network, thereby achieving significantly better anonymity guarantees while preserving Tor's throughput and latency.

Figure 1 depicts the result of applying our monitor to Tor's archive data, from Tor Metrics [29], of the last two years, where the anonymity guarantees are averaged for each day for the sake of readability. In the experiment the monitor assumes that 0.5% of all nodes are compromised, that the user requested the ports 443 (HTTPS) and 194 (IRC) and wants to compare itself against the set of users that solely request port 443, i.e., the probability that it is in the anonymity set including the set of users that solely surf[1]. The figure compares Tor's path selection algorithm against DISTRIBUTOR for sender anonymity and recipient anonymity and shows that DISTRIBUTOR leads to significant improvements. The figure additionally illustrates that a user's degree of anonymity highly fluctuates. This fluctuation underscores the value of providing a real-time, user-centric, anonymity monitor.

3 The extended AnoA Framework

The theoretical framework upon which we base MATOR is an extension of the ANOA framework for proving anonymity guarantees of anonymous communication networks [6]. ANOA is a useful building block for analyzing anonymity, but it lacks the generality to model the scenarios in MATOR. As a consequence, we extend the ANOA framework in this section such that it is suitable for proving that the anonymity bounds of our MATOR monitors are secure. In this section, we first present the extended ANOA challenger for stronger, adaptive adversaries and define adversary classes as realistic restrictions for this strong adversary. We then analyze how the anonymity guarantees provided by our extended ANOA framework compose under sequential composition. Finally, we present instantiations of the three anonymity notions we need for MATOR: sender anonymity, recipient anonymity and relationship anonymity.

3.1 The extended AnoA challenger

ANOA generalizes the notion of adjacency from differential privacy [10] to anonymity definitions. The challenger of ANOA defines at its heart an indistinguishability game in which a challenge bit b is chosen, and depending on this bit b one of two settings is executed. The goal of the adversary is to guess this bit b.

The challenger for ANOA (for n challenges) expects two kinds of messages: input messages of the form (input, m), that are processed by the protocol by running the protocol and letting a user $\mathcal{S}$ send the message m to a recipient $\mathcal{R}$, and challenge messages of the form $(\mathsf{challenge}, r_1, r_2, \Psi)$, where r_1 and r_1 correspond to two different inputs and where Ψ is an identifier for the challenge. In each challenge message $(\mathsf{challenge}, r_1, r_2, \Psi)$, the two messages r_1, r_2 have to be adjacent in a generalized sense. To simplify the notation of our adjacency functions, they get the bit b from the challenger as an additional input. Consequently, the adjacency functions can

---

[1] We choose port 443 as the Tor browser uses by default the add-on HTTPS Everywhere, which uses HTTPS whenever possible.

```
Adaptive ANOA Challenger $\textsc{Ch}(\mathcal{P}, \alpha, n, b)$

Upon message(input, $r = (\mathcal{S}, \mathcal{R}, m, \mathsf{sid})$)
  $\mathrm{RunProtocol}(r, 0)$

Upon message (challenge, $r_0, r_1, \Psi$)
  if $\Psi \notin \{1, \ldots, n\}$ then abort
  else if $\Psi \in \mathbb{T}$ then /* if $\Psi$ is known */
      Retrieve $\mathsf{s} := \mathsf{s}_\Psi$
      if $\mathsf{s} = \mathsf{over}$ then abort
  else $\mathsf{s} := \mathsf{fresh}$ and add $\Psi$ to $\mathbb{T}$ /* if $\Psi$ is fresh */
  Compute $(r^*, \mathsf{s}_\Psi) \leftarrow \alpha(\mathsf{s}, r_0, r_1, b)$
  $\mathrm{RunProtocol}(r, \Psi)$

RunProtocol($r = (\mathcal{S}, \mathcal{R}, m, \mathsf{sid}), \Psi$)
  if $\neg\exists y$ such that $(\mathsf{sid}, y, \Psi) \in S$ then /* if sid is fresh */
      Let $\mathsf{sid}_{\mathrm{real}} \leftarrow \{0,1\}^k$; Store $(\mathsf{sid}, \mathsf{sid}_{\mathrm{real}}, \Psi)$ in $S$.
  else $\mathsf{sid}_{\mathrm{real}} := y$ /* if a real session id for sid has already been chosen */
  Run $\mathcal{P}$ on $r = (\mathcal{S}, \mathcal{R}, m, \mathsf{sid}_{\mathrm{real}})$ and forward all messages that are sent by $\mathcal{P}$ to the adversary $\mathcal{A}$ and send all
  messages by the adversary to $\mathcal{P}$.
```

Figure 2: Extended ANOA Challenger

model the anonymity challenges on their own and simply output a message r^* that is sent to the challenger. We assume an anonymity function α, with the following interface:

$$\alpha(r_0 = (\mathcal{S}_0, \mathcal{R}_0, m_0, \mathsf{sid}_0), r_1 = (\mathcal{S}_1, \mathcal{R}_1, m_1, \mathsf{sid}_1), b)$$

For every challenge with tag Ψ, the challenger maintains a state s_Ψ that can be either fresh, over, or contain some information about an ongoing challenge. For every challenge that is not in the state over, we apply the anonymity function with its correct state – fresh for newly started challenges, some other state s_Ψ if the challenge is already active – and simulate the protocol on the output of the adjacency function. The challenger only allows n challenges. This restriction is implemented by restricting the set of possible *challenge tags* Ψ, which are used by the adversary to distinguish the challenges. We store all tags of already started challenges Ψ in a set $\mathbb{T}$.

To model more complex anonymity notions, such as anonymity for a *session*, the adjacency function is allowed to keep state for each challenge, i.e., sessions created by challenge messages are isolated from sessions created by input messages. Thus, an adversary cannot use the challenger to hijack sessions. This is done by choosing and storing session IDs $\mathsf{sid}_{\mathrm{real}}$ for every challenge separately as follows. Whenever a message with a new session ID sid is to be sent to the protocol, randomly pick a fresh session ID $\mathsf{sid}_{\mathrm{real}}$ that is sent instead, and store $\mathsf{sid}_{\mathrm{real}}$ together with sid and the challenge tag Ψ (if it is a challenge) or zero (if it is not a challenge). We store all mappings of sessions $(\mathsf{sid}, \mathsf{sid}_{\mathrm{real}}, \Psi)$ in a set S.

By definition, the sessions created by challenge messages are isolated from sessions created by input messages. Thus, an adversary cannot use the challenger to hijack sessions.

Multiplicative factor. ANOA introduces a multiplicative factor e^ε to the indistinguishability based definition, which gives the notion a flavor of differential privacy. Although they do not illustrate the usefulness of this factor in their analysis, we find that such a factor can in some cases be surprisingly helpful in describing anonymity guarantees. The most prominent case we have found is the analysis of an adversary that compromises no, or only a very limited amount of nodes. Before describing the details of this aspect we present our generalization of ANOA and then later elaborate on the impact of a multiplicative factor in Section 6.4.

The full description of the session challenger is available in Figure 2.

Definition 1 ((n, ε, δ)-α-IND-CDP)**.** *A protocol $\mathcal{P}$ is (n, ε, δ)-α-IND-CDP for a class of adversaries A, with $\varepsilon \geq 0$ and $0 \leq \delta \leq 1$, if for all PPT machines $\mathcal{A}$,*

$$\begin{aligned} &\Pr\left[0 = \langle A(\mathcal{A}(n)) || \textsc{Ch}(\mathcal{P}, \alpha, n, 0)\rangle\right] \\ &\leq e^{n\varepsilon} \Pr\left[0 = \langle A(\mathcal{A}(n)) || \textsc{Ch}(\mathcal{P}, \alpha, n, 1)\rangle\right] + e^{n\varepsilon} n\delta \end{aligned}$$

where the challenger Ch *is defined in Figure 2.*

Example 1: Single message sender anonymity. For modeling the simplest version of sender anonymity, where only a single message is sent by one of two possible senders, the adjacency function α_{SA} *depending on the challenge bit* b *simply chooses which of the two senders sends the message. As the adjacency function models sender anonymity, it makes sure that no information is leaked by the message itself or the recipients identity. Therefore it chooses the same message and the same recipient in both scenarios, e.g., by requiring that they are equal or by simply always choosing the message and recipient that are sent forthe first scenario. Moreover, since here we only allow a single message per session, for each message a fresh session is created and this session is terminated by setting the state to* over*, i.e.,* $\alpha_{\mathrm{SA}}(\mathsf{s}, (\mathcal{S}_0, \mathcal{R}_0, m_0), (\mathcal{S}_1, \_, \_), b) = ((\mathcal{S}_b, \mathcal{R}_0, m_0), \mathsf{over})$. *In Section 3.4 we formally define the anonymity functions for sender, recipient, and relationship anonymity.* ◇

Modifications made to AnoA. In ANOA only static, i.e., non-interactive, scenarios can be modeled, which excludes two-way communication. Moreover, the adversary is not restricted in its choices to determine the actions of the challenge users, which excludes many interesting and realistic scenarios. In practice, it makes an enormous difference to which server a user wants to connect, as this might require a different port or different settings. The worst-case adversary in ANOA could always choose the ports and settings for which the weakest guarantees can be given (c.f. Section 6 for the influence of ports and settings on anonymity). However, such an adversary is meaningless in practice because a user is rather interested in its real anonymity (e.g., the difficulty to distinguish its behavior from a regular user that only uses port 443 for HTTPS) and not in a superficial worst-case behavior.

3.2 Adversary classes

An adversary class $A(\cdot)$ is a wrapper that restricts the adversary $\mathcal{A}$ in its possible output behavior, and thus, in its knowledge about the world. Technically, it is a PPT machine $A(\mathcal{A})$ that internally runs the adversary $\mathcal{A}$ and forwards all messages that are sent from a compromised node to the adversary $\mathcal{A}$ and vice versa.

Example 2: Hiding recipients. With such a notion of an adversary class we can model an interactive scenario in which the adversary solely learns the information that an ISP would learn (and the information from its additionally compromised Tor nodes). The adversary class $A(\cdot)$ *computes the behavior of the servers without informing the adversary* $\mathcal{A}$ *about the communication. Thus, the adversary class would respond for the servers and the adversary will not be directly informed about the messages by the servers themselves. In the same way, the adversary class can additionally control which responses from the servers the adversary can see and, possibly, allow very few choices and solely inform the adversary whether a challenge session has ended.* ◇

The following example shows a scenario in which an (unrestricted) adaptive adversary might be too strong.

Example 3: Tor with entry guards. In ANOA *all actions, including the responses that servers give to users, are computed by the adversary. This design makes the definition of use Consider the Tor network with entry guards. Every user selects a small set of entry nodes (his guards) from which he chooses the entry node of every circuit. Guards are rotated only after several months. As a compromised entry node is fatal for the security guarantees that Tor can provide, the concept of entry guards helps in reducing the risk of choosing a malicious node. However, if such an entry guard is compromised the impact is more severe since an entry guard is used for a large number of circuits.*

An adaptive adversary can choose its targets adaptively and thus perform the following attack. It (statically) corrupts some nodes and then sends (polynomially) many messages $(\mathsf{input}, r = (\mathcal{S}, \_, \_, \_))$ *for different users* $\mathcal{S}$*, until one of them, say Alice, chooses a compromised node as its entry guard. Then* $\mathcal{A}$ *proceeds by using Alice and some other user in a challenge. As Alice will quite likely use the compromised entry guard again, the adversary wins with a very high probability (that depends on the number of guards per user).* ◇

Although this extremely successful attack is not unrealistic (it models the fact that some users that happen to use compromised entry guards *will* be deanonymized), it might not depict the attack scenario that we are interested in. Thus, we define an adversary class that makes sure that the adversary cannot choose its targets. Whenever the adversary starts a new challenge for (session) sender anonymity, the adversary class draws two users at random and places them into the challenge messages of this challenge.

Downward composability. The adversary classes that we define here are downwards composable for all protocols. More precisely, if a protocol $\mathcal{P}$ is α secure for an adversary class $A_1(\cdot)$ and if A_2 is an arbitrary adversary class, then $\mathcal{P}$ is also α secure for an adversary class $A_1(A_2(\cdot))$.

This observation follows directly from the fact that within the adversary classes, arbitrary PPT Turing machines are allowed, which includes wrapped machines $A_2(\cdot)$.

Complex anonymity properties as adversary classes. It is possible to describe complex anonymity properties as the composition of a simpler adjacency function and an adversary class. For modeling sender anonymity in the presence of specific user profiles, for example, we can simply compose an adversary class that models the user profile with the (session) sender anonymity adjacency function (c.f. Section 3.4).

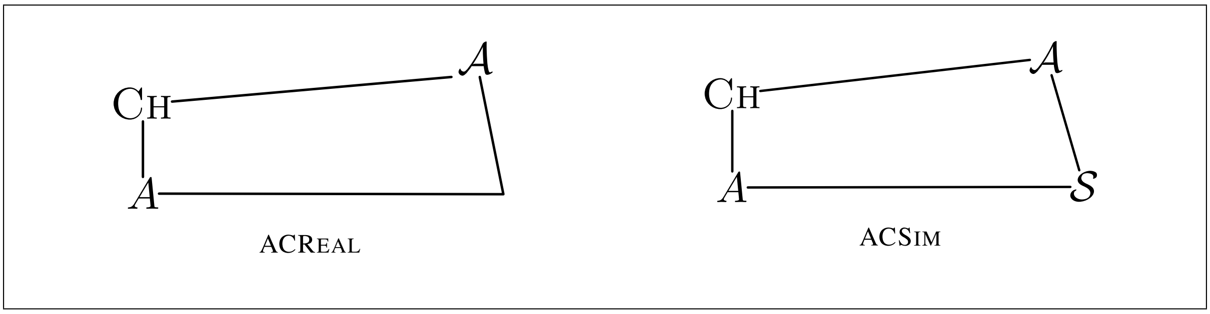

Figure 3: The two games from Construction 1

Defining user profiles as adversary classes. In [22], realistic, but relatively simple user profiles are defined that determine the behavior of users. We can model such profiles by defining a specific adversary class. The adversary class initiates profiles such as the *typical* users (they use, e.g., Gmail, Google Calendar, Facebook and perform web search at certain points in time) or *BitTorrent* users (they use Tor for downloading files on other times) as machines and measures time internally. If the adversary sends an input message, the adversary class initiates a profile. The adversary class might also initiate more profiles for random users at the beginning, which corresponds to "noise". If the adversary sends a challenge message, the adversary class initiates profiles for two different users (or two different profiles, depending on the anonymity notion).

On its own, the adversary class machine runs a loop that activates the profiles and increases a value t for time every now and then, until the adversary decides to halt with its guess b^* of the challenge bit b. Although the adversary class activates the profiles in a random order, it makes sure that all profiles have been activated before it proceeds to the next point in time. It additionally tells the activated profiles the point in time, such that they can decide whether or not they want to output a valid user action r or an error symbol $\perp$. The adversary class then sends input messages or challenge messages, depending on how the profiles have been initialized.

3.3 Sequential composability

It turns out that for some combinations of adversary classes and protocols proving an anonymity property for a single session does not imply that this anonymity property holds for multiple session, i.e., sequential composition does not hold. As an example, consider a protocol that leaks all secrets once the second message is sent, and consider an adversary class $A(\mathcal{A})$ that only forwards $\mathsf{challenge}$-messages to the challenger and that blocks all input-messages. This combination of a protocol and an adversary class is insecure only from the second challenge on; hence, for this combination sequential composability does not hold.

We show that sequential composability holds for adversary classes where additional challenges do not enable qualitatively new attacks, i.e., the adversary does not gain any conceptually new power. This can be ensured by requiring conditions from an adversary class $A(\mathcal{A})$ with respect to α.

Before we define these composability conditions, we first present a few helpful definitions. To discuss what it means that a challenge is simulatable, we consider the following two games, where ACREAL presents a real game with a regular adversary $\mathcal{A}$ within an adversary class $A(\mathcal{A})$ that interacts with a challenger: $\text{CH} \leftrightarrow A(\mathcal{A})$. In contrast, ACSIM presents a game in which part of what the adversary sends to the adversary class is simulated by a simulator $\mathcal{S}$ that is placed between the adversary and the adversary class: $\text{CH} \leftrightarrow A(\mathcal{S}(\mathcal{A}))$.

Construction 1. *Consider the following two scenarios (as in Figure 3):*

$\text{ACREAL}(b, n)$*:* $\mathcal{A}$ *communicates with* $A(\mathcal{A})$ *and* $\mathcal{A}$ *communicates with* $\text{CH}(b, n)$*. The bit of the challenger is* b *and the adversary may send challenge tags in* $\{1, \ldots, n\}$.

$\text{ACSIM}_{\mathcal{S}_{\mathbf{z}}}(b, n)$*:* $\mathcal{A}$ *communicates with* $\mathcal{S}_{\mathbf{z}}(b)$ *that in turn communicates with* $A(\mathcal{A})$ *and* $\mathcal{A}$ *communicates with* $\text{CH}(b, n)$*. The bit of the challenger is* b *and the adversary may send challenge tags in* $\{1, \ldots, n\}$.

We index a simulator with a *simulator index* $\mathbf{z} = [(z_1, b_1), \ldots, (z_n, b_n)] \in \{0, 1\}^{2n}$ that defines for which challenge tags this simulator simply forwards the messages and for which it simulates the challenge (and for which bit this challenge is simulated). If the bit z_i of the simulator index is set to sim, the simulator transforms the challenge with tag i, by simulating the adjacency function on the challenge message for the bit b_i.

The simulator indexes explicitly specify the challenge bit for which they simulate the adjacency function, as the simulator is unable to know the real challenge bit b. We call two simulator indexes *consistent with respect to* b, if for every challenge the simulator behaves consistently for both indexes (either does not simulate the challenge for both indexes or simulates them both for the same bit or simulates the adjacency function correctly).

Definition 2 (Consistent simulator index)**.** *A simulator index (for n challenges) is a bitstring* $\mathbf{z} = [(z_1, b_1), \ldots, (z_n, b_n)] \in \{0,1\}^{2n}$. *A pair of simulator indices* $\mathbf{z}, \mathbf{z}' \in \{0,1\}^{2n}$ *(for n challenges) is consistent w.r.t. b if*

$$\begin{aligned} \forall i \in \{1, \ldots, n\} \quad & \textit{s.t. } z_i = z'_i = \mathsf{sim}. \\ & b_i = b'_i \\ \textit{and } \forall i \in \{1, \ldots, n\} \quad & \textit{s.t. } z_i \neq z'_i. \\ & (z_i = \mathsf{sim} \Rightarrow b_i = b) \wedge (z'_i = \mathsf{sim} \Rightarrow b'_i = b). \end{aligned}$$

Equipped with the notation from above, we define the composability conditions for an adversary class. The conditions which we coin *reliability*, *alpha-renaming* and *simulatability* make sure that the security of a protocol is not broken qualitatively depending on quantitative changes. Reliability ensures that the adversary class does not initiate its own challenges. It may, however, still modify or drop challenges. Alpha-renaming ensures that the behavior of the adversary class is independent of the actual challenge tags per se and it does not interpret challenge tags semantically. Simulatability ensures that the adversary can simulate challenge messages (for a bit that he guessed) by input messages.

Definition 3 (Conditions for adversary class)**.** *An adversary class $A(\cdot)$ is* composable *for an anonymity function α, if the following conditions hold:*

1. ***Reliability:*** *$A(\mathcal{A})$ never sends a message $(\mathsf{challenge}, \_, \_, \Psi)$ to the challenger before receiving a message $(\mathsf{challenge}, \_, \_, \Psi)$ from $\mathcal{A}$ with the same challenge tag Ψ.*
2. ***Alpha-renaming:*** *$A(\mathcal{A})$ does not behave differently depending on the challenge tags Ψ that are sent by $\mathcal{A}$ except for using it in its own messages $(\mathsf{challenge}, \_, \_, \Psi)$ to the challenger and in the (otherwise empty) message $(\mathsf{answer\ for}, \_, \Psi)$ to $\mathcal{A}$.*
3. ***Simulatability:*** *For every $n \in \mathbb{N}$ and every simulator index $\mathbf{z} = [(z_1, b_1), \ldots, (z_n, b_n)] \in \{0,1\}^{2n}$ there exists a machine $\mathcal{S}_{\mathbf{z}}$ such that:*
    - *(a) For every $i \in \{1, \ldots, n\}$. If $z_i = \mathsf{sim}$ then $\mathcal{S}_{\mathbf{z}}$ never sends a message $(\mathsf{challenge}, \_, \_, i)$ to $A(\cdot)$.*
    - *(b) The games $\mathrm{ACR{\small EAL}}(b, n)$ and $\mathrm{ACS{\small IM}}_{\mathcal{S}_{\mathbf{z}_{\mathsf{dontsim}}}}(b, n)$ (cf. Construction 1) are computationally indistinguishable, where $\mathbf{z}_{\mathsf{dontsim}} = [(\mathsf{dontsim}, \_), \ldots, (\mathsf{dontsim}, \_)] \in \{0,1\}^{2n}$ for $\mathcal{S}_{\mathbf{z}}$ and $A(\mathcal{A})$.*
    - *(c) for all simulator indices $\mathbf{z}, \mathbf{z}' \in \{0,1\}^{2n}$ that are consistent w.r.t. b (see Definition 2) $\mathrm{ACS{\small IM}}_{\mathcal{S}_{\mathbf{z}}}(b, n)$ and $\mathrm{ACS{\small IM}}_{\mathcal{S}_{\mathbf{z}'}}(b, n)$ are indistinguishable.*

3.3.1 Composability theorem

Theorem 1. *For every protocol $\mathcal{P}$, every anonymity function α, every $n \in \mathbb{N}$ and every adversary class $A(\mathcal{A})$ that is composable for α. Whenever $\mathcal{P}$ is $(1, \varepsilon, \delta)$-α-IND-CDP for $A(\mathcal{A})$, with $\varepsilon \geq 0$ and $0 \leq \delta \leq 1$, then $\mathcal{P}$ is $(n, n \cdot \varepsilon, n \cdot e^{n\varepsilon} \cdot \delta)$-$\alpha$-IND-CDP for $A(\mathcal{A})$.*

Proof. We will show this theorem inductively. Assume that $\mathcal{P}$ is $(i, i \cdot \varepsilon, e^{i\varepsilon} \cdot i \cdot \delta)$-$\alpha$-IND-CDP. We show that $\mathcal{P}$ is also $(i+1, (i+1) \cdot \varepsilon, e^{(i+1) \cdot \varepsilon}(i+1) \cdot \delta)$-$\alpha$-IND-CDP.

Let $\mathcal{A}$ be an adversary that sends at most $i+1$ challenges. To do so, we construct several games:

- **Game:** G_0 is the normal game $\mathrm{ACR{\small EAL}}(0, i+1)$ with up to $i+1$ challenges where $b = 0$.
- **Game:** G_1 is an intermediate game $\mathrm{ACS{\small IM}}_{\mathcal{S}_{\mathbf{z}_{\mathsf{dontsim}}}}(0, i+1)$. Here every message from $\mathcal{A}$ to $A(\mathcal{A})$ (and otherwise) goes through the simulator $\mathcal{S}_{\mathbf{z}_{\mathsf{dontsim}}}$. However, this simulator does not need to simulate anything, as there are still up to $i+1$ challenges and $b = 0$.

  Claim: G_0 and G_1 are computationally indistinguishable.

  Proof: By item 3b Definition 3 the simulator $\mathcal{S}_{\mathbf{z}_{\mathsf{dontsim}}}$ exists and the games are indistinguishable.
- **Game:** G_2 is an intermediate (hybrid) game $\mathrm{ACS{\small IM}}_{\mathcal{S}_{\mathbf{z}}}(0, i+1)$ with $b = 0$ and fixed input messages instead of the challenge with tag $i+1$ (so there are at most i challenges left). This is done by using the simulator $\mathcal{S}_{\mathbf{z}}$ for $\mathbf{z} = [(\mathsf{dontsim}, \_), \ldots, (\mathsf{dontsim}, \_), (\mathsf{sim}, 0)] \in \{0,1\}^{i+1}$, i.e., the simulator simulates the $i+1$st challenge for $b = 0$.

  Claim: G_1 and G_2 are computationally indistinguishable.

  Proof: By item 3 of Definition 3, we know that the simulator $\mathcal{S}_{\mathbf{z}}$ exists. Since the simulator $\mathcal{S}_{\mathbf{z}}$ from G_2 uses the correct bit $b_{i+1} = 0$ for the simulated challenge, Item 3c of Definition 3 implies that the games are indistinguishable.

- **Game:** G_3 is the intermediate (hybrid) game $\textsc{ACSim}_{\mathcal{S}_{\mathbf{z}}}(1, i+1)$ where the simulator stays $\mathcal{S}_{\mathbf{z}}$ but the challenger changes to $b = 1$.

  Claim: G_2 and G_3 are $(i\varepsilon, e^{i\varepsilon} i\delta)$-indistinguishable.

  Proof: The adversary $\mathcal{S}_{\mathbf{z}}(\mathcal{A})$ makes at most i queries with challenge tags in $\{1, \ldots, i\}$. From the reliability property of the adversary class (item 1 of Definition 3) we know that thus $A(\mathcal{S}_{\mathbf{z}}(\mathcal{A}))$ uses at most i challenge tags in $\{1, \ldots, i\}$. The claim immediately follows from the induction hypothesis: $\mathcal{P}$ is $(i, i \cdot \varepsilon, i \cdot \delta)$-$\alpha$-IND-CDP.

- **Game:** G_4 is a game $\textsc{ACSim}_{\mathcal{S}_{\mathbf{z}'}}(1, i+1)$ where the simulator $\mathcal{S}_{\mathbf{z}'}$ with $\mathbf{z}' = [(\mathsf{sim}, 1), \ldots, (\mathsf{sim}, 1), (\mathsf{sim}, 0)]$ simulates all challenges from $\mathcal{A}$. For the challenge tags 1 to i, $\mathcal{S}_{\mathbf{z}'}$ simulates the challenges for $b_1 = \ldots = b_i = 1$, whereas for the tag $i + 1$ it still simulates it for $b_{i+1} = 0$. The challenger uses $b = 1$.

  Claim: G_3 and G_4 are computationally indistinguishable.

  Proof: Since the simulator $\mathcal{S}_{\mathbf{z}'}$ from G_4 uses the correct bit $b_1 = \ldots = b_i = 1$ for the challenges that are not simulated in $\mathcal{S}_{\mathbf{z}}$, Item 3c of Definition 3 implies that the games are indistinguishable.

- **Game:** G_5 is the game $\textsc{ACSim}_{\mathcal{S}_{\mathbf{z}'}}(0, i+1)$ where we use the same simulator $\mathcal{S}_{\mathbf{z}'}$ but we have $b = 0$ again.

  Claim: G_4 and G_5 are computationally indistinguishable.

  Proof: Since there are no challenge messages (everything is simulated, as by item 3a $\mathcal{S}_{\mathbf{z}'}$ does not send any messages $(\mathsf{challenge}, \_, \_, \Psi)$), changing the bit b of the challenger does not have any effect. Hence, the games are indistinguishable.

- **Game:** G_6 is the game $\textsc{ACSim}_{\mathcal{S}_{\mathbf{z}''}}(0, i+1)$ where we use the simulator $\mathcal{S}_{\mathbf{z}''}$ with $\mathbf{z}'' = [(\mathsf{sim}, 1), \ldots, (\mathsf{sim}, 1), (\mathsf{dontsim}, \_)]$. In other words, we do not simulate the challenge for $i + 1$ with $b_{i+1} = 0$, but we use the challenger again (also with $b = 0$).

  Claim: G_5 and G_6 are computationally indistinguishable.

  Proof: Since the simulator $\mathcal{S}_{\mathbf{z}'}$ from G_5 uses the correct bit $b_{i+1} = 0$ for the simulated challenge (which the simulator $\mathcal{S}_{\mathbf{z}''}$ does not simulate), Item 3c of Definition 3 implies that the games are indistinguishable.

- **Game:** G_7 is $\textsc{ACSim}_{translator(\mathcal{S}_{\mathbf{z}''})}(0, i+1)$ where we build around the simulator $\mathcal{S}_{\mathbf{z}''}$ we an interface $translator(\cdot)$ that translates the challenge tag from $i{+}1$ to 1 and vice versa in all messages $(\mathsf{challenge}, \_, \_, \Psi)$ from $\mathcal{S}_{\mathbf{z}''}$ to $A(\mathcal{A})$ and in all messages $(\mathsf{answer\ for}, \_, \Psi)$ from $A(\mathcal{A})$ to $\mathcal{S}_{\mathbf{z}''}$..

  Claim: G_6 and G_7 are information theoretically indistinguishable.

  Proof: Item 2 of Definition 3 requires that the renaming of challenge tags does not influence the behavior of $A(\mathcal{A})$. It also does not influence the behavior of the challenger (by definition) or the protocol (that never sees challenge tags). Thus, the games are indistinguishable.

- **Game:** G_8 is the game $\textsc{ACSim}_{translator(\mathcal{S}_{\mathbf{z}''})}(1, i+1)$ where the simulator is defined as in G_7 but $b = 1$.

  Claim: G_7 and G_8 are (ε, δ) indistinguishable.

  Proof: By assumption of the theorem, the protocol $\mathcal{P}$ is $(1, \varepsilon, \delta)$-α-IND-CDP for $A(\mathcal{A})$. Moreover, by definition of $\mathbf{z}''$ and by item 3a, the adversary $translator(\mathcal{S}_{\mathbf{z}''}(\mathcal{A}))$ only uses at most one challenge tag, namely the tag 1. From the reliability property of the adversary class (item 1 of Definition 3) we know that thus $A(translator(\mathcal{S}_{\mathbf{z}''}(\mathcal{A})))$ uses only the challenge tag 1. Thus, G_7 and G_8 are (ε, δ) indistinguishable.

- **Game:** G_9 is $\textsc{ACSim}_{\mathcal{S}_{\mathbf{z}''}}(1, i+1)$ where we remove the translation interface again.

  Claim: G_8 and G_9 are information theoretically indistinguishable.

  Proof: As before, Item 2 of Definition 3 requires that the renaming of challenge tags does not influence the behavior of $A(\mathcal{A})$. It also does not influence the behavior of the challenger (by definition) or the protocol (that never sees challenge tags). Thus, the games are indistinguishable.

- **Game:** G_{10} is the normal game $\textsc{ACReal}(1, i+1)$ where $b = 1$.

  Claim: G_9 and G_{10} are computationally indistinguishable.

  Proof: Since $\mathcal{S}_{\mathbf{z}''}$ uses the correct bit $b_1 = \ldots = b_i = 1$ for all simulations, we can replace it with $\mathcal{S}_{\mathbf{z}_{\mathsf{dontsim}}}$, that, in turn, is indistinguishable from $\textsc{ACReal}(1, i+1)$.

We slightly abuse notation in writing $\Pr[0 = \mathcal{A}(G_0)]$ for $\Pr[0 = \langle A(\mathcal{A}(n)) || \textsc{Ch}(\mathcal{P}, \alpha, n, 0)\rangle]$, $\Pr[0 = \mathcal{A}(G_1)]$ for $\Pr[0 = \langle A(\mathcal{S}_{\mathbf{z}}(b, \mathcal{A}(n))) || \textsc{Ch}(\mathcal{P}, \alpha, n, 0)\rangle]$, etc..

$$
\begin{aligned}
&\Pr[0 = \mathcal{A}(G_0)] \\
&\leq \Pr[0 = \mathcal{A}(G_1)] + \mu_1 \\
&\leq \Pr[0 = \mathcal{A}(G_2)] + \mu_2 + \mu_1 \\
&\leq e^{i\varepsilon} \Pr[0 = \mathcal{A}(G_3)] + e^{i\varepsilon} i\delta + \mu_2 + \mu_1 \\
&\leq e^{i\varepsilon} \Pr[0 = \mathcal{A}(G_4)] + e^{i\varepsilon}(\mu_3 + i\delta) + \mu_2 + \mu_1 \\
&\leq e^{i\varepsilon} \Pr[0 = \mathcal{A}(G_5)] + e^{i\varepsilon}(\mu_4 + \mu_3 + i\delta) + \mu_2 + \mu_1 \\
&\leq e^{i\varepsilon} \Pr[0 = \mathcal{A}(G_6)] + e^{i\varepsilon}(\mu_5 + \mu_4 + \mu_3 + i\delta) + \mu_2 + \mu_1 \\
&= e^{i\varepsilon} \Pr[0 = \mathcal{A}(G_7)] + e^{i\varepsilon}(\mu_5 + \mu_4 + \mu_3 + i\delta) + \mu_2 + \mu_1 \\
&\leq e^{i\varepsilon}(e^{\varepsilon} \Pr[0 = \mathcal{A}(G_8)] + \delta) + e^{i\varepsilon}(\mu_5 + \mu_4 + \mu_3 + i\delta) + \mu_2 + \mu_1 \\
&= e^{(i+1)\varepsilon} \Pr[0 = \mathcal{A}(G_8)] + e^{i\varepsilon}(\mu_5 + \mu_4 + \mu_3 + (i+1)\delta) + \mu_2 + \mu_1 \\
&= e^{(i+1)\varepsilon} \Pr[0 = \mathcal{A}(G_9)] + e^{i\varepsilon}(\mu_5 + \mu_4 + \mu_3 + (i+1)\delta) + \mu_2 + \mu_1 \\
&\leq e^{(i+1)\varepsilon} \Pr[0 = \mathcal{A}(G_{10})] + e^{(i+1)\varepsilon}\mu_6 + e^{i\varepsilon}(\mu_5 + \mu_4 + \mu_3 + (i+1)\delta) + \mu_2 + \mu_1 \\
&\leq e^{(i+1)\varepsilon} \Pr[0 = \mathcal{A}(G_{10})] + e^{(i+1)\varepsilon}(i+1)\delta
\end{aligned}
$$

□

3.4 Anonymity Notions

Finally, we are ready to present the anonymity notions for sender anonymity, recipient anonymity and relationship anonymity.

Sender anonymity. Sender anonymity models the scenario in which a malicious recipient, which might additionally control some Tor nodes, tries to determine the sender of a message. Since we are not interested in modeling the (semantic) privacy of messages, we model that the messages do not depend on the sender.

For ease of exposition, we first only allow challenges that consist of one single message each. We can model this simple variant of sender anonymity, called *Single Message Sender Anonymity*, as follows:

```
$\alpha_{\mathrm{SA}}(\mathsf{s}, r_0 = (\mathcal{S}_0, \mathcal{R}_0, m_0), r_1 = (\mathcal{S}_1, \_, \_), b)$
  output $((\mathcal{S}_b, \mathcal{R}_0, m_0), \mathsf{over})$
```

Note that the state of the challenge is set to over. For challenges that span a whole session (consisting of several messages) we require that within this session the sender does not change (i.e., the adversary cannot choose a new pair of potential senders per message, but only per challenge). Subsequently, we define the anonymity function for *Session Sender Anonymity* as follows.

```
$\alpha_{\mathrm{SSA}}(\mathsf{s}, r_0 = (\mathcal{S}_0, \mathcal{R}_0, m_0, \_), r_1 = (\mathcal{S}_1, \_, \_, \_), b)$
  if $\mathsf{s} = \mathsf{fresh} \lor \mathsf{s} = (\mathcal{S}_0, \mathcal{S}_1)$ then
    output $((\mathcal{S}_b, \mathcal{R}_0, m_0, 1), \mathsf{s} := (\mathcal{S}_0, \mathcal{S}_1))$
```

Recipient Anonymity. In recipient anonymity the adversary is assumed to control the link to the Tor network, e.g., by compromising the ISP of the user. The goal of the adversary is thus to find out which web pages a user visits. Similar to sender anonymity, the adversary additionally controls some Tor nodes.

Recall that in Example 2, adversary classes are needed for modeling recipient anonymity for representing the servers and potentially the user profiles. Hence, recipient anonymity can only be proven for certain adversary classes. In addition to requiring that the servers are machines and the users follow certain user profiles, as in Example 2, there is an inherent insecurity for recipient anonymity.

Various so-called website fingerprinting attacks [8, 18, 28] are known against the anonymous communication service Tor that directly break recipient anonymity. In these attacks, the adversary recognizes recipient by their traffic patterns, such as direction changes or size. These attacks, however, are not specific to Tor. Every low-latency anonymous channel is prone to these kinds of attacks. Hence, for a settings in which such website fingerprinting attacks are possible, recipient anonymity cannot hold.

As a consequence, we only consider settings in which such attacks are not possible. As a first step, we define an anonymity function α_{RA} for recipient anonymity that excludes challenge messages that are of different size.

Similar to sender anonymity, we model the requirement that the sender is constant by always choosing the sender $\mathcal{S}_0$ from the first message.

```
$\alpha_{\mathrm{SRA}}(\mathsf{s}, r_0 = (\mathcal{S}_0, \mathcal{R}_0, m_0, \_), r_1 = (\_, \mathcal{R}_1, m_1, \_), b)$
  if $(\mathsf{s} = \mathsf{fresh} \lor \mathsf{s} = \mathcal{S}_0) \land |m_0| = |m_1|$ then
    output $((\mathcal{S}_0, \mathcal{R}_b, m_b, 1), \mathsf{s} := \mathcal{S}_0)$
```

These restrictions in the anonymity function, however, do not suffice. We can only define recipient anonymity for servers, i.e., recipients, in which the response of the recipients has the same length for all messages that are of the same length, i.e., the length of the response of a server solely depends on the length of the message.

Relationship Anonymity. In relationship anonymity the adversary is an observer that might control some Tor nodes and that tries to deanonymize a communication between a sender and a recipient. As both sender and recipient of this communication are previously unknown to the adversary (otherwise relationship anonymity collapses to either sender anonymity or recipient anonymity), we require a slightly more complex anonymity function:

```
$\alpha_{\mathrm{SRel}}(\mathsf{s}, r_0 = (\mathcal{S}_0, \mathcal{R}_0, m_0, \_), r_1 = (\mathcal{S}_1, \mathcal{R}_1, \_, \_), b)$
  if $\mathsf{s} = \mathsf{fresh}$ then
    $a \leftarrow \{0, 1\}$
  else if $\exists x.\ \mathsf{s} = (\mathcal{S}_0, \mathcal{S}_1, x)$ then
    $a := x$
  if b=0 then
    output $((\mathcal{S}_a, \mathcal{R}_a, m_0, 1), \mathsf{s} := (\mathcal{S}_0, \mathcal{S}_1, a))$
  else
    output $((\mathcal{S}_a, \mathcal{R}_{1-a}, m_0, 1), \mathsf{s} := (\mathcal{S}_0, \mathcal{S}_1, a))$
```

In this function there are four possible scenarios for a challenge session: Each of the senders could send its messages to each recipient and again the choice is made depending on the bit of the challenger. If $b = 0$, then one message is sent, by a randomly chosen sender $\mathcal{S}_a$ (that is then used for the whole challenge) to the recipient $\mathcal{R}_a$ that was specified for this sender by the adversary. If $b = 1$, the (still randomly chosen) sender $\mathcal{S}_a$ sends its messages to the other recipient $\mathcal{R}_{1-a}$. The goal of the adversary is to figure out whether one of its challenges was chosen or whether the combination of sender and recipient was swapped.

4 Modeling Tor in AnoA

We briefly recall how Tor and its path selection work and then we explain how they are modeled.

4.1 The Tor's path selection (PSTOR) algorithm

As mentioned in Section 2, nodes on circuits (or paths) in Tor are *not* selected (uniform) randomly. To improve the performance, Tor's current path selection algorithm makes a weighted random choice over all nodes that support the user's connections and preferences, and bases the weights on information that is retrieved from a periodically published *server descriptor* and an hourly published *consensus document*. These documents are generated and maintained by a small set of semi-trusted directory authorities, and contain up-to-date information about each node.

In a server descriptor, a Tor node publishes its bandwidth, the ports it would allow as an exit node, its so-called *family* (used for distributing trust), its uptime, its operating system, and its version of Tor. In order to prevent malicious Tor nodes from equivocating (i.e., sending different information to different users), the nodes are required to periodically upload (every 12 to 18 hours) their current server descriptor to all directory authorities.

The consensus document is computed hourly by the directory authorities, and it contains for each node information such as the node's availability, its entrusted bandwidth, a pointer to the up-to-date server descriptor, and whether this node should be used as an exit node and/or an entry node. Moreover, the consensus document contains entry, middle, and exit scaling factors for every node in order to balance the bandwidth. This scaling is necessary since there are fewer nodes that are marked as exit nodes (∼1000 in May 2014) than as entry (∼4000 in May 2014) or middle nodes (∼5000 in May 2014).

The PSTOR algorithm computes the weight of a node based on the retrieved node's entrusted bandwidth. Since a circuit establishment is expensive, the path selection tries to include as many of the requested ports into one circuit as possible. Given a list of requested ports by the user, PSTOR determines the maximal set of ports that is supported by any exit node, and then excludes all nodes that do not support this maximal set of ports and that are not marked as exit nodes. Then, the algorithm assigns a weight to every remaining $node$ by dividing its entrusted

```
$\mathrm{exitProb}(ex, [\mathcal{N}, pf, c])$
if ($ex$ had an Exit tag in the latest consensus) $\wedge$ ($ex$ is suited for $c$ and for $pf$) then
    Let bS be an empty list
    for $node \in \mathcal{N}$ do
        if $node$ offers more ports from $c$ than the elements in bS then
            $\mathrm{bS} :=$ new list, only containing $node$
        else if $ex$ offers exactly the same ports as the elements in bS then
            $\mathrm{bS} := \mathrm{bS}.append(node)$
    if $ex$ offers $|\mathrm{bS}|$ many ports then
        $\mathrm{totalSupport} := 0$
        for $node \in \mathrm{bS}$ do
            $\mathrm{totalSupport} := \mathrm{totalSupport} + scEx(node)$
        return $scEx(ex)/\mathrm{totalSupport}$
    else return $0$
else return $0$
```

Figure 4: The computation of the exit node probabilities for Tor's path selection algorithm

bandwidth $node.bw$ by the sum of the entrusted bandwidths s of all not excluded nodes and multiplies this with the corresponding exit scaling factor $scEx(node)$ from the consensus document: $node.bw/s * scEx(node)$. Finally, the algorithm performs a weighted random choice over these nodes.

As Tor is built upon the principle of distributing trust, the path selection excludes circuits with nodes that are *related*, i.e., that are in the same /16 subnet and nodes that are in each other's family. After having chosen the exit node, the path selection chooses an entry node in two steps: first, the algorithm excludes all nodes that are related to the exit node and all nodes that are not marked as entry nodes in the current consensus; second, the algorithm computes the weight of each of the remaining nodes by dividing their entrusted bandwidth by the sum of all not excluded nodes and performs a weighted random choice over these nodes. For the middle node the path selection proceeds as for the entry nodes except that middle nodes do not require specific tags. However, all relatives of both the exit node and the entry node are excluded.

This path selection algorithm adapts to the preferences of the user, who can, e.g., decide to only use nodes that have the 'stable' tag or to build circuits that only use 'fast' nodes. Tor's path selection algorithm also offers a configuration for including non-valid entry or exit nodes as well as entry nodes that are not considered to be entry guards.

4.2 The Tor protocol in extended AnoA

We base our model of Tor on our previous work that models Tor as a UC protocol Π_{OR} [5]. Π_{OR} is based on Tor's specification and accurately models the key exchange, the circuit establishment, and the process of relaying message cells over Tor.

However, Π_{OR} abstracts from Tor's path selection by considering a uniform path selection. In our work, we use an extension of Π_{OR}, where instead of the uniform path selection the above described PSTOR algorithm is used. This extension of Π_{OR} gives us a protocol on which we can base our analysis of MATOR.

This extension of Π_{OR}, which we call ${\Pi_{\text{OR}}}'$, solely extends the path selection algorithm in Π_{OR} and leaves everything else untouched. We accordingly extend the ideal functionality $\mathcal{F}_{\text{OR}}$ from [5], which abstracts from all cryptographic operations in Π_{OR}, with Tor's actual path selection. $\mathcal{F}_{\text{OR}}$ uses a shared memory between the honest parties and sends handles over the network instead of the onion ciphertexts that Tor sends over the network. Each party looks up then which message corresponds to the respective handle. In this way, the adversary does not learn more than the handles about the single messages that are sent over the network. We call the extension of the ideal functionality $\mathcal{F}_{\text{OR}}$ with Tor's actual path selection ${\mathcal{F}_{\text{OR}}}'$. Since Π_{OR} and $\mathcal{F}_{\text{OR}}$ both execute the same path selection algorithm, the UC realization proof for Π_{OR} and $\mathcal{F}_{\text{OR}}$ applies verbatim to ${\Pi_{\text{OR}}}'$ and ${\mathcal{F}_{\text{OR}}}'$ (Theorem 1).

In [5] *secure OR modules* are defined, which comprise a one-way authenticated key exchange protocol, and secure encryption and decryption operations. Moreover, that work uses a network functionality $\mathcal{F}_{\text{NET}_q}$ for modeling partially global network adversaries that compromise at most q links, a key registration functionality $\mathcal{F}_{\text{REG}}$ and a functionality for a confidential and mutually authenticated channel (for modeling HTTPS connections) $\mathcal{F}_{\text{SCS}}$.

Proposition 1. *[c.f. [5]] If $\Pi_{\mathrm{OR}}{}'$ uses secure OR modules M, then with the global functionality $\mathcal{F}_{\mathrm{NET}_q}$ the resulting protocol $\Pi_{\mathrm{OR}}{}'$ in the $\mathcal{F}_{\mathrm{REG},\mathrm{SCS}}$-hybrid model securely realizes the ideal functionality $\mathcal{F}_{\mathrm{OR}}{}'$ for any q.*

5 Anonymity Monitors

In this section, we estimate these anonymity bounds of Tor for a given snapshot of the Tor network computed from a given consensus document. We devise anonymity monitors that a client can run along with the Tor client to estimate sound anonymity bounds for the PSTOR algorithm. First, we present the anonymity monitor for sender anonymity that gives a bound for the probability that a malicious server manages to deanonymize the user. Second, we present the anonymity monitor for recipient anonymity that bounds the probability that the user's ISP can determine the recipient of a communication. Third, we present the anonymity monitor for relationship anonymity that bounds the probability that an observer can relate traffic to a sender and a recipient.

The output of these three monitors provides a bound of the maximal success probability of a worst-case adversary to deanonymize a Tor user at a given point in time. The success probability is calculated for an individual circuit creation, i.e., whenever a new circuit is created the adversary may deanonymize the user for this circuit with the success probability.

Adversary model. For our monitors, we consider a worst-case adversary that statically compromises a given number k of Tor nodes of its choice. It may use any auxiliary information it possesses; however, we assume that it is has no prior knowledge of the entry guards a user will choose or has chosen. In practice the adversary can gather its k nodes by attacking the existing nodes or (rather easily) by adding new nodes to the Tor network.

5.1 Modeling MATor in AnoA

In MATOR we analyze the anonymity of a real Tor user depending on her preferences pf and connections conn and the verified Tor consensus file and the latest server descriptors.

Technically, we instantiate MATOR in ANOA by defining an adversary class for the user that restricts the ANOA adversary as follows. As we are interested in the anonymity of a specific user, we overwrite the adversary's choice of preferences for the challenge users with the ones from the real user. The adversary may still choose the preferences of other users in the network

Definition 4 (MATOR profile)**.** *For a set of user preferences pf and two sets of ports $\mathsf{ports}_1, \mathsf{ports}_2$ the profile $\mathrm{MATOR}_{(pf,\mathsf{ports}_1,\mathsf{ports}_2)}$ replaces all messages of the form*

$$\begin{aligned} &(\mathsf{challenge},(\mathcal{S}_1, \bot, (\mathsf{initialize}, \_, \_), \mathsf{sid}_1),\\ &\qquad(\mathcal{S}_2, \bot, (\mathsf{initialize}, \_, \_), \mathsf{sid}_1), \Psi)\\ \textit{with } &(\mathsf{challenge},(\mathcal{S}_1, \bot, (\mathsf{initialize}, pf, \mathsf{ports}_1), \mathsf{sid}_1),\\ &\qquad(\mathcal{S}_2, \bot, (\mathsf{initialize}, pf, \mathsf{ports}_2), \mathsf{sid}_2), \Psi)\end{aligned}$$

and by blocking all challenge messages in which only one user sends $\mathsf{initialize}$*. Consequently, only the preferences pf and ports* ports_1 *and* ports_2 *that are specified for the profile can be used for challenges.*

In our monitors' implementations, we prepare the anonymity analysis as follows.

Monitor—Preparing the scenario. The monitor first prepares the scenario, consisting of user preferences pf, a list of connections conn that describes to which servers (particularly over which ports) the user wants to connect, and information about the current consensus file and the respective server descriptors. The monitor then uses a simulation of the PSTOR algorithm to compute the weights of all nodes for these connections, each depending on the possible position of the node within the circuit (entry, middle, exit) and depending on the node-specific data from the consensus file (e.g., its tags, bandwidth and IP address).

The details of how these weights are computed heavily depend on the path selection that is to be used. A short definition of how the path selection algorithm for Tor computes the probability of a given exit node is given in Figure 4, where $scEx()$ weights a node depending on its bandwidth, its tags, and the weights given in the consensus.

5.2 Computing Sender Anonymity Bounds

Sender anonymity considers an adversary that wants to deanonymize a Tor user communicating to a given server that is under adversarial control.

```
$\text{SAMonitor}(\mathcal{N}, pf, ps)$
for $ex \in \mathcal{N}$ do
    for $en \in \mathcal{N}$ do
        if $\text{ps}.allows(en, ex, pf)$ then
            $exitP := \text{ps.exP}(ex)$; $entryP = \text{ps.enP}(en, ex)$
            $\delta(en)$+=$exitP \cdot entryP$
sort all nodes in $\mathcal{N}$ by their $\delta(\cdot)$ value in a list $sorted$
for $node \in sorted$ and $1 \leq i \leq k$ do
    $\delta$ += $\delta(node)$; $i := i + 1$
return $\delta$
```

Figure 5: Sender Anonymity Monitor

A long history of prior work on traffic correlation attacks shows that whenever an adversary controls the entry and the exit node of a circuit, or the entry node and the server, it may deanonymize the user [9, 12, 17, 20, 21, 24–26]. Directly following the analysis from [6] we notice that a (local) adversary with control over the server can deanonymize the user only if it manages to compromise the entry node of a circuit. Our monitor thus checks the probability that an entry node is compromised, which depends on the user's preferences and connections as well as on the active nodes. We give guarantees even against a worst-case adversary that compromises the k most probable entry nodes.

Figure 5 depicts the pseudocode for computing the bounds for sender anonymity. Here, the adversary compromises the k most likely entry nodes for the given connection. Since the probability that an entry node is chosen is independent of the user (for a fixed connection), the sender anonymity bound is the sum that any of these nodes is chosen. Thus, the bounds computed by the sender anonymity monitor are the worst case bounds.

Theorem 2 (Sender Anonymity Monitor)**.** *Given a consensus and a corresponding set of server descriptors, let $\mathcal{N}$ be a set of nodes with bandwidth weights, the preferences pf of the user, the ports* ports *to which the user wants to connect, and a path selection algorithm ps that uses these informations. Then for the output δ of the algorithm* $\text{SAMonitor}(\mathcal{N}, pf, ps)$ *the following holds: against passive local adversaries $\Pi_{\text{OR}}{}'$ satisfies $(1, 0, \delta) - \alpha_{\text{SSA}}$-IND-CDP$_{\text{MATOR}_{(pf,\text{ports},\emptyset)}}$, where α_{SSA}-IND-CDP$_{\text{MATOR}_{(pf,\text{ports},\emptyset)}}$ denotes session sender anonymity (see Section 3) with the* MATOR *profile as in Definition 4.*

Proof. By Theorem 1 and Lemma 22 from the full version of the AnoA framework [7, Lemma 22], we know that it suffices to show that $(1, 0, \delta) - \alpha_{\text{SSA}}$-IND-CDP$_{\text{MATOR}_{(pf,\text{ports},\emptyset)}}$ holds for the ideal functionality $\mathcal{F}_{\text{OR}}{}'$.

Recall that the ideal functionality $\mathcal{F}_{\text{OR}}{}'$ sends handles over the network instead of onions (i.e. ciphertexts) for honest nodes. For compromised nodes, $\mathcal{F}_{\text{OR}}{}'$ reveals which handles belong together, and if all nodes of the circuit to the exit node are compromised, $\mathcal{F}_{\text{OR}}{}'$ additionally reveals the message along with the handle.

Let A_b denote the event that A in the game b correctly guesses the bit b and A_{1-b} denote the event that A in the game $1 - b$ wrongly guesses the bit b, and let C_b denote the event that the circuit C is chosen by the path selection in the game b. Since only the user differs in a sender anonymity scenario, i.e., the preferences and the connections are the same, by the construction of Tor's path selection algorithm (see Figure) the distribution of selected path is the same for both user connections:

$$\Pr[C_b] = \Pr[C_{1-b}]$$

Hence, we omit the subscript b for C_b in the following.

Since the sender is the same in both scenarios and since the same messages are sent over the circuit, the view of the adversary is the same once the circuit is fixed and the adversary did not compromise the circuits entry node. We say that a circuit is *honest*, written as H_C, if for the circuit $C = (n_1, n_2, n_3)$ the entry node n_1 is not compromised. Hence, for all circuits C, the following holds

$$\Pr[A_b \mid C \wedge H_C] = \Pr[A_{1-b} \mid C \wedge H_C] \tag{1}$$

Observe that the events C and H_C are independent since the adversary first chooses which nodes to compromise, and the path selection does not know which nodes are compromised.

Observe that

$$Pr[A_b] = \sum_{C \in nodes^3} \Pr[A_b \mid C] \cdot \Pr[C] \tag{2}$$

Moreover, observe that

$$\begin{aligned}
&Pr[A_b \mid C] \\
=\ & \Pr[A_b \mid C \wedge H_C] \cdot \Pr[H_C] \\
&+ \underbrace{\Pr[A_b \mid C \wedge \neg H_C]}_{=1} \cdot \Pr[\neg H_C] \\
\overset{(1)}{=}\ & \Pr[A_{1-b} \mid C \wedge H_C] \cdot \Pr[H_C] + \Pr[\neg H_C] \\
=\ & \Pr[A_{1-b} \mid C \wedge H_C] \cdot \Pr[H_C] \\
&+ \underbrace{0}_{=\Pr[A_{1-b} \mid C \wedge \neg H_C]} \cdot \Pr[\neg H_C] + \Pr[\neg H_C] \\
=\ & \Pr[A_{1-b} \mid C \wedge H_C] \cdot \Pr[H_C] \\
&+ \Pr[A_{1-b} \mid C \wedge \neg H_C] \cdot \Pr[\neg H_C] + \Pr[\neg H_C] \\
=\ & Pr[A_{1-b} \mid C] + \Pr[\neg H_C]
\end{aligned} \tag{3}$$

Thus, we have the following:

$$\begin{aligned}
&\Pr[A_b] \\
\overset{(2)}{=}\ & \sum_{C \in nodes^3} \Pr[A_b \mid C] \cdot \Pr[C] \\
\overset{(3)}{=}\ & \sum_{C \in nodes^3} (\Pr[A_{1-b} \mid C] + \Pr[\neg H_C]) \cdot \Pr[C] \\
=\ & \sum_{C \in nodes^3} \Pr[A_{1-b} \mid C] \cdot \Pr[C] + \Pr[\neg H_C] \cdot \Pr[C] \\
=\ & \Pr[A_{1-b}] + \underbrace{\sum_{C \in nodes^3} \Pr[\neg H_C] \cdot \Pr[C]}_{=:\delta_A}
\end{aligned}$$

Let A be the adversary that compromises the entry nodes with the k-highest weights for the set of ports ports.

Claim 1 (A is the most successful adversary)**.** *For each adversary A' that compromises at most k nodes the following holds: $\delta_{A'} \leq \delta_A$.*

Proof of Claim 1. Recall that in a sender anonymity scenario the connections and the preferences are the same in both games. Hence, all entry nodes have the same weights in both games. By the construction of Tor's path selection, the weight of an entry node equals the overall probability with which the node is chosen as an entry node. In other words:

$$entryW(n) = \sum_{(n_2,n_3) \in unrelatedNodes^2} ps.enP(n, n_3) \cdot ps.miP(n_2, n, n_3) \cdot ps.exP(n_3)$$

As a consequence, A compromises those entry node n such that the sum

$$\sum_{(n_2,n_3) \in \mathcal{N}^2} \Pr[(n, n_2, n_3)]$$

is k-maximal, i.e., is among the k highest sums. Let K be the k entry nodes with these highest sums.

Then, for any other adversary A' we have the following:

$$\begin{aligned}
\delta_{A'} &= \sum_{C \in nodes^3} \Pr[\neg H_C] \cdot \Pr[C] \\
&\leq \sum_{\substack{(n_2,n_3) \in nodes^2 \\ n \in K}} \Pr[(n, n_2, n_3)] = \delta_A
\end{aligned}$$

$\diamond$

$\square$

The sender anonymity monitor and the guard mechanism. The sender anonymity monitor outputs a bound on the probability that the adversary might deanonymize a user's communication using Tor. This deanonymization is heavily based on the probability of compromising the user's entry node, and thus, sender anonymity only changes whenever a fresh entry node is chosen. If the user makes use of the guard mechanism [27, 34] she will not choose a fresh entry node for every communication, but only choose from a small set of entry nodes that is kept for a long time.

The guard mechanism modifies the adversaries possibilities to deanonymize the user as follows: As the user rarely choses a fresh entry node it is less likely that the adversary manages to compromise the entry node, even if the user builds many circuits. However, if it manages to compromise a user's guard, it will be able to deanonymize this user very often and consistently until she changes her guards again.

5.3 Computing Recipient Anonymity Bounds

Recipient anonymity considers the setting with a malicious ISP of the user that wants to find out which web pages a user visits and additionally controls some Tor nodes. As presented earlier, in AnoA recipient anonymity is formalized by comparing two challenge settings in which the user is the same but the recipients differ.

For this analysis we exclude website fingerprinting attacks, as those attacks are attacks on the content of messages and present an orthogonal attack vector that is not related to the communication protocol. A malicious ISP could always additionally host fingerprinting attacks, independent of the protocol we analyze. Technically, we assume that the adversary cannot learn information from the answers to challenge messages.[2] Recently, some provably secure defenses have been developed against these fingerprinting attacks [33].

For computing the recipient anonymity bound, the probability of three scenarios has to be considered. The first and most relevant scenario encompasses compromising the exit node in the user's circuit, where recipient anonymity is immediately broken. The second scenario considers a compromised middle node: in this case, the adversary knows the circuit's exit node and can check whether this exit node does not offer a port that is requested in one of the settings. The third scenario considers a compromised entry node: even in this case, the adversary can learn which middle node is chosen in the circuit and thereby check whether this middle node is more probable in one of the settings, e.g., because the middle node is related to a heavy-weighted (i.e., very probable) exit node that does not offer one of the ports in one of the two challenge settings. Additionally, the adversary might learn something by seeing the entry node that has been chosen, e.g., it might be less probable to choose an entry node that is related to a very heavy-weighted exit node.

A precise estimation of the recipient anonymity bounds has to take into account how much an adversary could learn from any given node and then add these values up for the k most advantageous nodes. For each $node$ and each of the three positions (entry, middle, and exit), the increase in the distinguishing probability has to be computed. For the exit position, the adversary can immediately distinguish the challenge settings by compromising the exit node; hence this increase δ_{Ex} is the probability that the node n is chosen as an exit node: $\delta_{Ex}(node) := node.bw/(\sum_{node' \in \mathrm{bS}_i} node'.bw)$.

For the entry position, the adversary can not directly learn which setting is chosen, but it can gain evidence that one of the settings is chosen, e.g., because some entry nodes are more probable to be chosen as entry nodes in the first setting since in the other settings these entry nodes are in the family of a heavy-weighted exit node that is only in the best support of the second setting. Hence, this increase can be characterized by the difference in the probabilities for every triple of nodes.

$$\delta_{En}(node) = \sum_{\substack{node_1 \in \mathrm{bS}_1 \\ node, node_1 \\ \text{unrelated}}} \; \sum_{\substack{node_2 \in \mathcal{N} \\ node, node_1, node_2 \\ \text{unrelated} \\ node, node_1, node_2 \\ \text{distinguishing}}} w'_1 - w'_2$$

where $node, node_1, node_2$ distinguishing $\Leftrightarrow$ $w'_1/w'_2 > e^{\varepsilon}$, and where $w'_i = \mathrm{ps.enP}(node, node_1) \cdot \mathrm{ps.exP}_i(node_1) \cdot \mathrm{ps.miP}(node_2, node, node_1)$.[3]

Iterating over all node triples, however, is not sufficiently efficient for a live recipient anonymity monitor. To improve the efficiency, we over-approximate the influence of the exit node on $\delta_{En}(node)$ and omit the exit nodes in the computation.

---

[2] For Tor this formally presents a form of length-regularity: the lengths of the responses the two possible recipients of a challenge message give have to be equally distributed for input messages of equal length.

[3] We calculate a bound on δ for a given multiplicative factor of $\varepsilon \geq 0$. Intuitively, we take a bound for ε as input and compute the probability that this bound cannot be achieved.

```
$\mathrm{RAMonitor}(\mathcal{N}, pf, \mathsf{ports}_1, \mathsf{ports}_2, ps, \varepsilon)$
for $ex$ in $\mathcal{N}$ do
    if $\mathrm{ps}.allows(ex, pf, \mathsf{ports}_1)$ then
        $\delta(ex)$+=$\mathrm{ps.exP}(exit, pf, \mathsf{ports}_1)$;
        for $en$ in $\mathcal{N}$ do
            if $\mathrm{ps}.allows(en, ex, pf, \mathsf{ports}_1)$ then
                $\delta_{sEn}$+=$\mathrm{ps.enP}(en, ex, pf, \mathsf{ports}_1) \cdot \mathrm{ps.exP}(ex, pf, \mathsf{ports}_1) - \mathrm{ps.enP}(en, ex, pf, \mathsf{ports}_2) \cdot \mathrm{ps.exP}(ex, pf, \mathsf{ports}_2)$
for $en$ in $\mathcal{N}$ do
    $tmp := 0$
    for $mi$ in $\mathcal{N}$ do
        $w_1 := \mathrm{ps.miPMaxEx}(mi, en, pf, \mathsf{ports}_1)$
        $w_2 := \mathrm{ps.miPMinEx}(mi, en, pf, \mathsf{ports}_2)$
        if $\frac{w_1}{w_2} > e^{\varepsilon}$ then
            $tmp$+=$(w_1 - w_2) \cdot \mathrm{ps.enPMaxEx}(en, pf, \mathsf{ports}_1)$
    $\delta(en)$+=$\min\{tmp, en.used\_as\_entry\}$
for $ex$ in $\mathcal{N}$ do
    if $\mathrm{ps}.allows(ex, pf, \mathsf{ports}_1)$ then
        for $mi$ in $\mathcal{N}$ do
            $w_1 := \mathrm{ps.miPMaxEn}(mi, ex, pf, \mathsf{ports}_1)$
            $w_2 := \mathrm{ps.miPMinEn}(mi, ex, pf, \mathsf{ports}_2)$
            if $\frac{w_1}{w_2} > e^{\varepsilon}$ then
                $\delta(mi)$+=$(w_1 - w_2) \cdot \mathrm{ps.exP}(ex, pf, \mathsf{ports}_1)$
sort all nodes $n$ by the value $\delta(n)$ in a list $sorted$
for $node$ in $sorted$ and $1 \leq i \leq k$ do
    $\delta$+=$\delta(node)$; $i = i + 1$
$\delta$+=$\delta_{sEn}$
return $\delta$
```

Figure 6: Recipient Anonymity Monitor

These approximations influence relevant but rather minor factors of the anonymity guarantee, as the δ mostly depends on the probability to compromise exit nodes. In the next section (Section 5.4) we will give a detailed description of our approximations.

Theorem 3 (Recipient Anonymity Monitor)**.** *For a consensus document, a set of server descriptors, a user's preferences pf and two sets of ports ports_1 and ports_2, let $\mathcal{N}$ be the set of nodes in the consensus together with their weights. Let $\varepsilon > 0$ be a real value . Then for the output δ of the algorithm* $\mathrm{RAMonitor}(\mathcal{N}, pf, \mathsf{ports}_1, \mathsf{ports}_2, ps, \varepsilon)$ *the following holds against passive local adversaries:*

$$\Pi_{\mathrm{OR}}' \text{ satisfies } (1, \varepsilon, \delta) - \alpha_{\mathrm{SRA}}\text{-IND-CDP}_{\mathrm{MATOR}(pf, \mathsf{ports}_1, \mathsf{ports}_2)},$$

where $\alpha_{\mathrm{SRA}}\text{-IND-CDP}_{\mathrm{MATOR}(pf, \mathsf{ports}_1, \mathsf{ports}_2)}$ denotes session recipient anonymity (see Section 3) .

Proof. By Theorem 1 and Lemma 22 from the full version of the AnoA framework [7, Lemma 22], we know that it suffices to show that $(1, \varepsilon, \delta) - \alpha_{\mathrm{SRA}}\text{-IND-CDP}_{\mathrm{MATOR}(pf, \mathsf{ports}_1, \mathsf{ports}_2)}$ holds for the ideal functionality $\mathcal{F}_{\mathrm{OR}}'$.

Recall that the ideal functionality $\mathcal{F}_{\mathrm{OR}}'$ sends handles over the network instead of onions (i.e. ciphertexts) for honest nodes. For compromised nodes, $\mathcal{F}_{\mathrm{OR}}'$ reveals which handles belong together, and if all nodes of the circuit to the exit node are compromised, $\mathcal{F}_{\mathrm{OR}}'$ additionally reveals the message along with the handle.

For a given ε, we can bound δ as follows, where k is the number of estimated compromised nodes from the user preferences pf.

$$\delta \leq \max_{K s.t. |K| \leq k} \sum_{n \in K} \delta(n)$$

where $\delta(n)$ is the increase in the advantage that the adversary gets from compromising the node n.

$$\delta(n) \leq \delta_{En}(n) + \delta_{Mi}(n) + \delta_{Ex}(n)$$

The recipient anonymity monitor approximates $\delta(n)$ for efficiency reasons. We show in the following that $\delta(n)$ is safely over-approximated. We first show that $\delta_{En}(n)$ is safely over-approximated, then we show that $\delta_{Mi}(n)$ is safely over-approximated, and for δ_{Ex} we do not need to show anything since it is precisely computed as

$$\delta_{Ex}(n) := n.bw / (\sum_{n' \in \mathrm{bS}_i} n'.bw).$$

For the sake of convenience, we omit in the following the preferences and ports if we call the path simulator functions ps. (See Figure 8 for the precise arguments for the call.) We say that a triple of nodes n, n_1, n_2 is *distinguishing* $\iff$ $w_1'/w_2' > e^{\varepsilon}$, and where $w_i' = \mathrm{ps.enP}(n, n_1) \cdot \mathrm{ps.exP}_i(n_1) \cdot \mathrm{ps.miP}(n_2, n, n_1)$. Then, we can over-approximate $\delta_{En}(n)$ as follows:

$$\begin{aligned}\delta_{En}(n) &= \sum_{\substack{n_1 \in \mathrm{bS}_1 \\ n, n_1 \\ \text{unrelated}}} \sum_{\substack{n_2 \in \mathcal{N} \\ n, n_1, n_2 \\ \text{unrelated} \\ n, n_1, n_2 \\ \text{distinguishing}}} w_1' - w_2' \\ &\leq \sum_{\substack{n_2 \in \mathcal{N} \\ n, n_2 \text{ unrelated} \\ n, n_2 \text{ quasi entry distinguishing}}} \mathrm{ps.enPMaxEx}(n) \cdot (w_1 - w_2)\end{aligned}$$

where n, n_2 quasi entry distinguishing $\iff$ $w_1/w_2 > e^{\varepsilon}$, and $w_1 := \mathrm{ps.miPMaxEx}(n_2, n)$ and $w_2 := \mathrm{ps.miPMinEx}(n_2)$. We stress that we have to use $\mathrm{ps.enPMaxEx}(n)$ because we do not know which exit node was chosen; hence we have to assume that the entry node with the maximal family has been chosen, whose impact is worst.

For $\delta_{Mi}(n)$, we also have

$$\begin{aligned}\delta_{Mi}(n) &= \sum_{\substack{n_1 \in \mathrm{bS}_1 \\ n, n_1 \\ \text{unrelated}}} \sum_{\substack{n_2 \in \mathcal{N} \\ n, n_1, n_2 \\ \text{unrelated} \\ n, n_1, n_2 \\ \text{distinguishing}}} w_1' - w_2' \\ &\leq \sum_{\substack{n_2 \in \mathcal{N} \\ n, n_2 \text{ unrelated} \\ n, n_2 \text{ quasi exit distinguishing}}} \mathrm{ps.exP}(n_2) \cdot (w_1 - w_2)\end{aligned}$$

where n, n_2 quasi exit distinguishing $\iff$ $w_1/w_2 > e^{\varepsilon}$, where $w_1 := \mathrm{ps.miPMaxEn}(n, n_2)$ and $w_2 := \mathrm{ps.miPMinEn}(n, n_2)$. We stress that for $\mathrm{ps.exP}(n_2)$ we can take the exact weight of the exit node because the choice of the exit node does not depend on the entry or middle node.

Note that our approximations are exactly what the recipient anonymity monitor computes. Hence, the bounds computed by the recipient anonymity monitor are secure. ☐

The recipient anonymity monitor and the guard mechanism. In recipient anonymity we assume that the adversary has knowledge about the user's connection to the Tor network. For our bounds we model both the case where the user does not use entry guards, i.e., uses a fresh entry node for every circuit and the case in which the user uses guards. To this end we allow the user to specify her entry guards (we envision that they are read from the Tor source-code if MATOR is integrated into Tor). In this case we allow the adversary to even have auxiliary information about the guards as we can assume that they have been used before. This in particular allows the adversary to target the entry guards if that helps it to break recipient anonymity. Note, however, that the entry node plays only a minor role in breaking recipient anonymity.

5.4 Approximations

In this section we explain the approximations we did for the recipient anonymity monitor (c.f. Section 5.3).

We approximate the probability that an entry node n chosen by always assuming that the exit node with the largest family has been chosen: $\mathrm{ps.enPMaxEx}(n, pf, \mathsf{ports}_1)$ (see Figure 7). Similarly, we over-approximate the conditional probability that a middle node is chosen in the first scenario, given an entry and exit node, by again assuming the exit node with the maximal family weight: $w_1 := \mathrm{ps.miPMaxEx}(n_2, n)$ (see Figure 7) . In order to have a sound approximation, we under-approximate the conditional probability that a middle node is chosen in the second scenario by not subtracting the weight of any exit family at all: $w_2 := \mathrm{ps.miPMinEx}(n_2)$ (see Figure 7).

Assume all variables are initialized with 0.

Computing the largest family joins

```
/* largest exit family join: compute for each entry node $en$ the cumulative middle weight of the largest family of valid exit nodes if $en$ is chosen as an entry node */
for each entry node $en$ in $\mathcal{N}$ do
    Compute the cumulative middle weight of the largest family join of $en$ with exit nodes $ex \in \mathrm{bS}_1$ such that $\mathrm{ps}.allow(en, ex, pf)$.
    Save this to $en.\mathrm{lExFJ}$
/* largest other exit family: compute for each entry node the cumulative entry weight of the largest family of valid exit nodes if en is chosen as an entry node */
for each entry node $en$ in $\mathcal{N}$ do
    Compute the cumulative entry weight of the largest family for exit nodes $ex \in \mathrm{bS}_1$ such that $\mathrm{ps}.allow(en, ex, pf)$.
    Save this to $en.\mathrm{lOEF}$
/* largest entry family join: compute for each exit node the cumulative middle weight of the largest family of valid exit nodes if ex is chosen as an exit node */
for each exit node $ex$ in $\mathcal{N}$ do
    Compute the cumulative middle weight of the largest family join (i.e., the node $en$ such that the family join is maximal) of $ex$ with entry nodes $en$ such that $\mathrm{ps}.allow(en, ex, pf)$.
    Save this to $ex.\mathrm{lEnFJ}$.
```

```
$\mathrm{miPMaxEx}(mi, en, pf, \mathsf{ports})$
    return $\frac{\mathrm{ps}.miBW(mi, pf, \mathsf{ports})}{total - en.\mathrm{lExFJ}}$

$\mathrm{miPMinEx}(mi, en, pf, \mathsf{ports})$
    return $\frac{\mathrm{ps}.miBW(mi, pf, \mathsf{ports})}{total - \mathrm{ps.familyMiddleWeight}(en, pf, \mathsf{ports})}$

$\mathrm{enPMaxEx}(en, pf, \mathsf{ports})$
    return $\frac{\mathrm{ps}.enBW(en)}{total - en.\mathrm{lOEF}}$

$\mathrm{miPMaxEn}(mi, ex, pf, \mathsf{ports})$
    return $\frac{\mathrm{ps}.miBW(mi, pf, \mathsf{ports})}{total - ex.\mathrm{lEnFJ}}$

$\mathrm{miPMinEn}(mi, ex, pf, \mathsf{ports})$
    return $\frac{\mathrm{ps}.miBW(mi, pf, \mathsf{ports})}{total - \mathrm{ps.familyMiddleWeight}(ex, pf, \mathsf{ports})}$
```

Figure 7: Tor Path Selection Approximations

For approximating whether a circuit is distinguishing or not for an entry node n and a middle node n_2, we check whether $w_1/w_2 > e^{\varepsilon}$.

$$\sum_{\substack{n_2 \in \mathcal{N} \\ n, n_2 \text{unrelated} \\ n, n_2 \text{distinguishing}}} \mathrm{ps.enPMaxEx}(n) \cdot (w_1 - w_2)$$

For the middle position, the adversary can in some cases learn which setting is chosen by identifying a port that is not offered by the observed exit node yet requested in one of the settings. For a node n, computing the increase $\delta_{Mi}(n)$ in the adversary's distinguishing probability is analogous to the increase for an entry node. Similar to $\delta_{En}(n)$, we over-approximate the effect of the entry node $\delta_{Mi}(n)$ as depicted in Figure 8.

Finally, we incorporate the effect of the adversary's ability to always observe which entry node is chosen. Recall that the probability for choosing a particular entry node can differ if different exit nodes are in the best support for different lists of requested ports in the two settings. To respect this impact on anonymity we add for each entry node n the difference of the probabilities that n is chosen in one setting and that n is chosen in the other setting:

$$\delta_{sEn}(n) := \sum_{ex \in \mathrm{bS}_1} \mathrm{enP}_1 - \mathrm{enP}_2,$$

where $\mathrm{enP}_i = \mathrm{ps.enP}(n, ex, pf, \mathsf{ports}_i)$.

We approximate the probabilities for relationship analogously.

5.5 Relationship Anonymity Bounds

Relationship anonymity considers a setting in which an observer tries to deanonymize a communication between a sender and a recipient. The adversary does not already control the ISP of the sender or the server, but corrupts or

```
$\mathrm{RelMonitor}(\mathcal{N}, pf, \mathsf{ports}_1, \mathsf{ports}_2, ps, \varepsilon)$
for $ex$ in $\mathcal{N}$ do
    if $\mathrm{ps}.allows(ex, pf, \mathsf{ports}_1)$ then
        for $en$ in $\mathcal{N}$ do
            if $\mathrm{ps}.allows(en, ex, pf, \mathsf{ports}_1)$ then
                $\delta_{EnEx}(en, ex) := \mathrm{ps.enP}(en, ex, pf, \mathsf{ports}_1) \cdot \mathrm{ps.exP}(ex, pf, \mathsf{ports}_1)$
                $\mathrm{enP}(en) += \delta_{EnEx}(en, ex)$
        for $mi$ in $\mathcal{N}$ do
            $w_1 := \mathrm{ps.miPMaxEn}(mi, ex, pf, \mathsf{ports}_1)$
            $w_2 := \mathrm{ps.miPMinEn}(mi, ex, pf, \mathsf{ports}_2)$
            if $\frac{w_1}{w_2} > e^{\varepsilon}$ then
                $\delta_{intermediate}(mi) += \mathrm{ps.exP}(ex, pf, \mathsf{ports1}) \cdot (w_1 - w_2)$
for $en$ in $\mathcal{N}$ do
    if $\mathrm{enP}(en) > 0$ then
        $t := 0$
        for $mi$ in $\mathcal{N} \setminus \{en\}$ do
            $w_1 := \mathrm{ps.miPMaxEx}(mi, en, pf, \mathsf{ports}_1)$
            $w_2 := \mathrm{ps.miPMinEx}(mi, en, pf, \mathsf{ports}_2)$
            if $\frac{w_1}{w_2} > e^{\varepsilon}$ then
                $t += \mathrm{ps.enPMaxEx}(en, pf, \mathsf{ports1}) \cdot (w_1 - w_2)$
            $\delta_{EnMi}(en, mi) += \mathrm{enP}(en) \cdot \delta_{intermediate}(mi)$
    $t := \min t, \mathrm{enP}(en)$
    $\delta_{En}(en) := t$
Compute (approximate) the respective $\delta$ for the maximal subset of nodes
```

Figure 8: Relationship Anonymity Monitor

controls some Tor nodes. As presented earlier, in ANOA, relationship anonymity is formalized by comparing two challenge settings in which there are two possible senders and two possible recipients. As for recipient anonymity, we exclude fingerprinting attacks for this analysis, as those attacks are based on the content of messages.

For computing the relationship anonymity bound we first observe that the adversary has to break the anonymity of the sender in order to succeed. Consequently, the scenarios in which an adversary gains an advantage are a combination of the scenarios from sender anonymity and from recipient anonymity: As long as the probability for choosing entry nodes is not different for different senders, the entry node has to be compromised. Still, the probability of three scenarios associated with recipient anonymity has to be considered. The most intuitive and most severe scenario is a compromised entry node in combination with a compromised exit node. In this case an adversary can immediately see both the sender and the recipient of a message, and thus, break relationship anonymity. The second scenario is a compromised entry node in combination with a compromised middle node. Here, depending on the ports that are used, an observed exit node might be more likely in one of the scenarios or even impossible, in some other scenarios. The third scenario is a compromised entry node without other compromised nodes. Even now the adversary might learn something by seeing the middle node, which might be more or less likely to be chosen, depending on which exit node was chosen (which, again, might depend on the scenario).

Similarly to recipient anonymity, computing a precise bound for relationship anonymity directly is infeasible for a light-weight anonymity monitor. Consequently we give a sound approximation for the anonymity guarantee. We give estimates for the advantage of the adversary for each combination of nodes, as well as for each individual entry node. To overcome the necessity to compute the best subset of all nodes such that their combinations yield the optimal advantage for the adversary, we observe that by compromising k nodes an adversary can only compromise at most $\frac{k\cdot(k-1)}{2}$ combinations of nodes. Our bound comprises of the top k values from compromising entry nodes only and the top $\frac{k\cdot(k-1)}{2}$ values that arise from combinations of two nodes.

Theorem 4 (Relationship Anonymity Monitor). *For a consensus document, a set of server descriptors, a user's preferences pf and two sets of ports* ports_1 *and* ports_2, *let $\mathcal{N}$ be the set of nodes in the consensus together with their weights. Let $\varepsilon > 0$ be a real value . Then for the output δ of the algorithm* $\mathrm{RelMonitor}(\mathcal{N}, pf, \mathsf{ports}_1, \mathsf{ports}_2, ps, \varepsilon)$ $\Pi_{\mathrm{OR}}{}'$ *satisfies* $(1, \varepsilon, \delta) - \alpha_{\mathrm{SRel}}\text{-IND-CDP}_{\mathrm{MATOR}(pf, \mathsf{ports}_1, \mathsf{ports}_2)}$ *against passive local adversaries, where* $\alpha_{\mathrm{SRel}}\text{-IND-CDP}_{\mathrm{MATOR}(pf, \mathsf{ports}_1, \mathsf{ports}_2)}$ *denotes session relationship anonymity (see Section 3).*

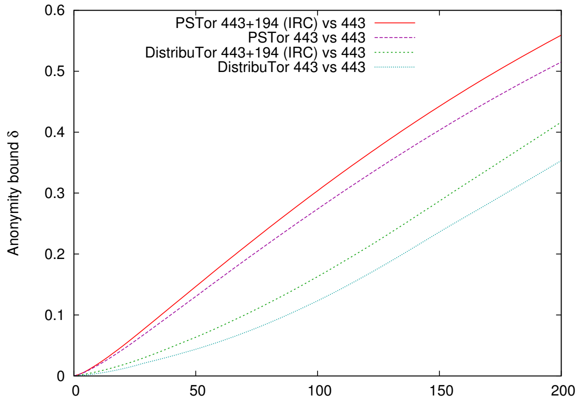

Figure 9: Relationship anonymity guarantees (value of δ) depending on the number of compromised nodes. We used $\varepsilon = 0$ and as settings the ports HTTPS+IRC vs. HTTPS.

Proof. The proof assumes that there is no distinguishing exit node, no distinguishing middle node and no distinguishing combination of exit and middle nodes. This holds for both PSTOR and DISTRIBUTOR.

By Theorem 1 and Lemma 22 from the full version of the AnoA framework [7, Lemma 22], we know that it suffices to show that $(1, \varepsilon, \delta) - \alpha_{\mathrm{SRel}}\text{-IND-CDP}_{\mathrm{MATOR}_{(pf, \mathsf{ports}_1, \mathsf{ports}_2)}}$ holds for the ideal functionality $\mathcal{F}_{\mathrm{OR}}{}'$.

In contrast to both sender anonymity and recipient anonymity, the impact that compromising a node has on δ highly depends on other nodes, as both sender and recipient have to be deanonymized. For the third scenario that we described above (only the entry node is compromized) we can compute the impact of every node n on δ, describe it as $\delta(n)$ and then sum over the k nodes with the largest values $\delta(n)$. For the other two scenarios (Entry and Exit node are compromized; entry and middle node are compromized), we instead compute how much impact a combination of two nodes has on δ and coin this impact for nodes n and m as $\delta_{EnEx}(n, m)$ and $\delta_{EnMi}(n, m)$ respectively.

For given ϵ, we can bound δ as follows, where k is the number of compromised nodes, and x is the number of compromised connections between Entry-Exit or Entry-Middle nodes:

$$\delta \leq \max_{K\ s.t. |K| \leq k} \sum_{n \in K} \left(\delta_{En}(n) + \sum_{m \in K} \delta_{EnEx}(n, m) + \delta_{EnMi}(n, m) \right)$$

To increase the efficiency of our monitors we safely approximate this computation by regarding the values for δ_{EnEx} and δ_{EnMi} as weights on a directed graph between nodes. We then collapse the graph into an undirected graph $G = (N, E)$ with nodes N and edges E. If k nodes within this graph are compromised, their total contribution to δ (in terms of δ_{EnEx} and δ_{EnMi}) is given by the sum over the weights δ_{edge} of all edges that connect them. Furthermore we see that for k nodes not more than $\frac{k(k-1)}{2}$ edges e can be compromised and thus can approximate the value for δ as:

$$\delta \leq \max_{K\ s.t. |K| \leq k} \sum_{n \in K} \delta_{En}(n) + \max_{X \subseteq E\ s.t. |X| \leq \frac{k(k-1)}{2}} \sum_{\mathbf{e} \in X} \delta_{edge}(\mathbf{e})$$

where for every $\mathbf{e} = \{n, m\}$, $\delta_{edge}(\mathbf{e}) = \delta_{EnEx}(n, m) + \delta_{EnEx}(m, n) + \delta_{EnMi}(n, m) + \delta_{EnMi}(m, n)$. We show in the following that these values are safely over-approximated, beginning with δ_{En}. We say that a triple en, mi, ex is *distinguishing* $\iff w'_1/w'_2 > e^{\epsilon}$, where $w'_i = ps.exP_i(ex) \cdot ps.enP(en, ex) \cdot ps.miP(mi, en, ex)$. Then we can over-approximate $\delta_{En}(en)$ as follows:

$$\begin{aligned} \delta_{En}(en) &= \sum_{\substack{ex \in bS_1 \\ ex, en \text{ unrelated}}} \quad \sum_{\substack{mi \in nodes \\ en, mi, ex \text{ unrelated} \\ en, mi, ex \text{ distinguishing}}} w'_1 - w'_2 \\ &\leq \sum_{\substack{mi \in nodes \\ en, mi \text{ unrelated} \\ en, mi \text{ quasi entry distinguishing}}} ps.enPMaxEx(en) \cdot (w_1 - w_2) \end{aligned}$$

where en, mi quasi entry distinguishing $\iff w_1/w_2 > e^{\epsilon}$ and $w_1 := ps.miPMaxEx(m, n)$, $w_2 := ps.miPMinEx(m, n)$. We stress that we have to use ps.enPMaxEx(n) because we do not know which exit node

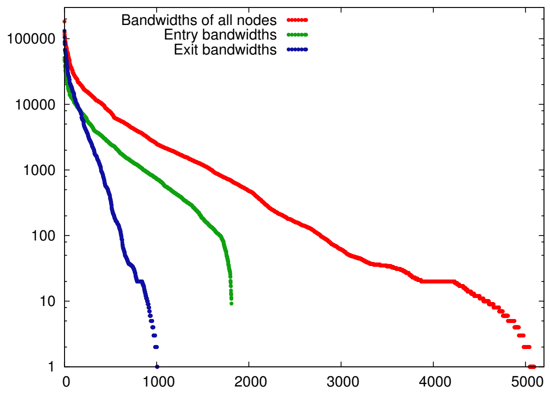

Figure 10: Bandwidth and weights chosen by Tor (from top to bottom: total bandwidths, entry bandwidths, exit bandwidths, one point for each node; the nodes are sorted by bandwidth for each line)

was chosen, so we choose the entry with the maximal family has been chosen whose impact is worst.

For approximating δ_{edge} we proceed as follows. The values for δ_{EnEx} can be computed directly by using the Path Selection algorithm:

$$\delta_{EnEx}(en, ex) = ps.exP_1(ex) \cdot ps.enP(en, ex)$$

We approximate the values for δ_{EnMi} analogously to the approximation of δ_{En} from above:

$$\begin{aligned}\delta_{intermediate}(mi) = & \sum_{\substack{ex \in bS_1 \\ ex,mi \text{ unrelated}}} \sum_{\substack{en \in nodes \\ en,mi,ex \text{ unrelated} \\ en,mi,ex \text{ distinguishing}}} w_1' - w_2' \\ \leq & \sum_{\substack{ex \in nodes \\ mi,ex \text{ unrelated} \\ mi,ex \text{ quasi exit distinguishing}}} ps.exP(ex) \cdot (w_1 - w_2)\end{aligned}$$

where mi, ex quasi exit distinguishing $\iff$ $w_1/w_2 > e^\epsilon$, $w_1 := ps.miPMaxEn(mi, en)$ and $w_2 := ps.miPMinEn(mi, en)$. As this is only middle node weight, to again avoid cubic-time computations , we over-approximate the probability that particular Entry-Middle pair is chosen by again assuming that the exit with the maximal family has been chosen:

$$\delta_{EnMi}(en, mi) = \delta_{intermediate}(mi) \cdot ps.enPMaxEx(en)$$

Note that our approximations are exactly what our relationship anonymity monitor computes. Hence, the bounds computed by the relationship anonymity monitor are secure. □

6 Experimental results

Our anonymity monitors described in Section 5 allow us to perform an analysis on the real consensus-data published by Tor [29]. In this section we present a selection of the guarantees that the monitors computed.

6.1 Implementation and Data collection

We implement our sender, recipient, and relationship anonymity monitors as multi-threaded C++ programs. The code comprises of approximately 3000 lines codes and employs SQLite [2] and Boost [3] libraries. The monitor programs are available on our website [1].

For our analysis we process the server descriptions of Tor that are released every month to construct a database of family relationships between nodes. Processing the server descriptors takes a significant amount of time (around 15 minutes), but we require this computation only once per month. Moreover, the information does not depend on the settings of the user, which means that the database could be precomputed and downloaded once per month.

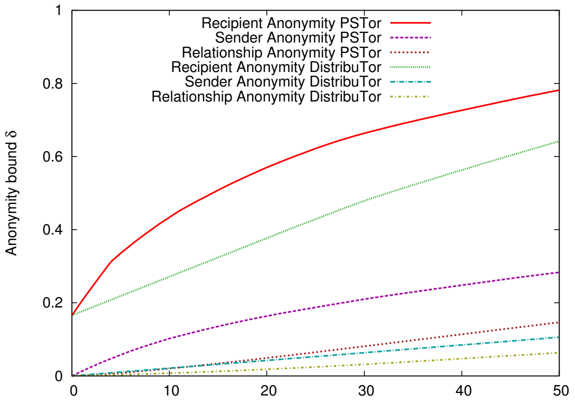

Figure 11: Comparison between PSTOR and DISTRIBUTOR. The graph shows the value for δ with $\varepsilon = 0$ for 0 to 50 compromised nodes of the adversary's choice.

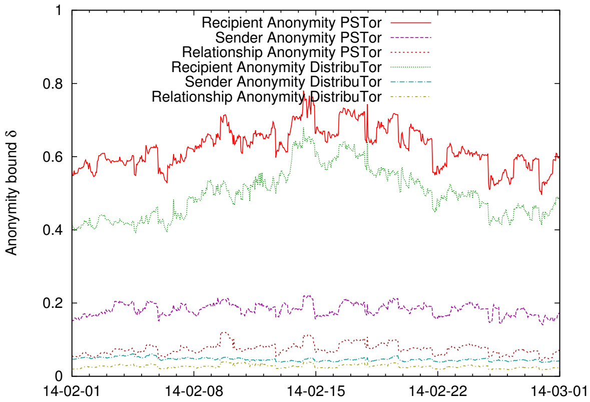

Figure 12: Anonymity guarantees (value of δ) over the course of February 2014 for 0.5% compromised nodes. We used $\varepsilon = 0$ and as settings the ports HTTPS+IRC vs. HTTPS.

Performance. We measured the performance of the monitors on a standard notebook (MacBook Air 2 GHz Intel Core i7, 4 GB 1600 MHz DDR3 RAM). Our monitors start by processing a consensus file and by computing the weights of all nodes depending on the path selection algorithm, the connections of the user (and of the scenario we want to compare the user with). In our performance evaluation we called this part of the computation "preparing the weights". Afterwards the anonymity guarantees are computed. The exact computation times are given in Figure 13 (averaged over 100 runs on consensus files from Feb. 2014).

6.2 Path Selection Strategies

The bandwidth of Tor nodes is not uniformly distributed as Tor tries improve its performance by selecting nodes depending on their bandwidth. As a result, a node with twice the bandwidth is twice as likely to be used for a circuit. The real-life bandwidth distribution, however, contains nodes that are several hundred times as likely as other nodes. (See the log-scale graph in Figure 10.) Consequently, a small number of nodes with a very high bandwidth is used in a large percentage of circuits. If these nodes get compromised or similar new nodes are added

Preparing the weights	3.39 sec.
Computing sender anonymity guarantee	0.73 sec.
Computing recipient anonymity guarantee	6.07 sec.
Computing relationship anonymity guarantee	9.10 sec.

Figure 13: Performance of our anonymity monitors

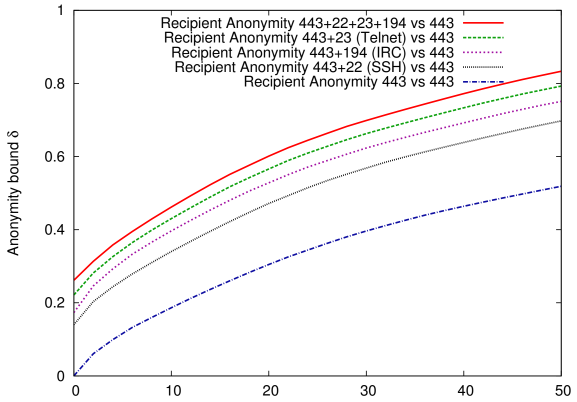

Figure 14: Impact of the choice of ports on recipient anonymity. We chose $\varepsilon = 0$. The graph depicts the value for δ depending on the number of compromised nodes (0 to 50). We compared all settings against HTTPS only (port 443).

to the network by an adversary, this adversary can deanonymize many connections. Consequently the current path selection of Tor produces obvious targets such that an attacker that compromises these points can deanonymize a significant part of the network.

Novel loss-less path selection: Re-balancing the workload. To reduce the risk posed by such high bandwidth nodes we propose DISTRIBUTOR, a path selection algorithm that distributes the trust amongst exit and entry nodes as evenly as possible. We observe that the exit bandwidth inherently is a bottleneck as only few nodes wish to be used as exit nodes. Consequently, we first focus on the exit nodes.

1. **Distributing the bandwidth for exit nodes:** We compute the exit bandwidth that Tor supports at a given moment by summing over the bandwidth of all possible exit nodes and weighting them according to their tags and the weights from the consensus. We then compute how evenly we can distribute the bandwidth by using small exit nodes solely as exit nodes and restricting the usage of the largest exit nodes accordingly. In this process we make sure that the previous exit bandwidth is still provided by our improved path selection algorithm.
2. **Distributing the bandwidth for entry nodes:** After the weight of nodes for their position as an exit node has been set we compute the weights for entry nodes. We proceed just as before, by trying to preserve the entry bandwidth and still distributing the trust in entry nodes as widely as possible.

Anonymity improvement. As we put a bound for the maximal weight of exit and entry nodes, we use their remaining bandwidth by increasing their weight to be used as middle node, as this position is considered least critical. The details of our redistribution can be found in Figure 15. In the following section we present experimental results computed on the real consensus data of Tor and evaluate DISTRIBUTOR against Tor's path selection algorithm.

Naturally it would be possible to sacrifice performance of the Tor network for a much stronger improvement in anonymity by reducing the targeted total bandwidth. In an extreme case one could weight all nodes uniformly, which would allow much stronger anonymity guarantees.

Note that we did not consider the case that the entry bandwidth poses a bottleneck for Tor. In this case, one should change the order in which these calculations are made.

6.3 Lessons learned

Advantages of DistribuTor over PSTor. As expected, our DISTRIBUTOR algorithm significantly improves sender anonymity and also moderately improves recipient anonymity in all experiments. The only moderate improvement of recipient anonymity is to be expected as the exit bandwidth inherently is a bottleneck of Tor. The improvement in sender anonymity, however, is more significant (see Figure 11). As the re-balancing path selection algorithm does not affect Tor's overall performance, it presents a possibility to improve the anonymity guarantees at virtually no cost.

Change in the anonymity guarantees over time. Our monitors also allow to analyze whether and how the anonymity guarantees vary over time. Such variations are the result of changes in the number of available servers,

```
$node.exitBW$
if $node$ can be used as exit then
    if $node.bw < maxExitBW$ then
        return $node.bw$
    else return $maxExitBW$
else return 0

$node.entryBW$
if $node$ can be used as entry then
    if $node$ can be used as exit then
        if $node.bw < maxExitBW$ then
            return 0
        else
            if $node.bw - maxExitBW < maxEntryBW$ then
                return $node.bw - maxExitBW$
            else return $maxEntryBW$
    else return 0
else return 0

$node.middleBW$
$bw := node.bw$
if $node$ can be used as exit then
    $bw := bw - maxExitBW$
if $node$ can be used as entry then
    $bw := bw - maxEntryBW$
if $bw > 0$ then return $bw$
else return 0
```

Figure 15: DistribuTor: Our redistribution of the bandwidths

their bandwidth and their exit policies. Figure 12 shows how the guarantees change over the course of a month (February 2014).

Anonymity guarantees over the last years. As a long-time study analyzed the guarantees for the last 24 Months in Figure 1 (c.f. Section 2). We smoothed the graph by computing the average anonymity for each day in order to improve the readability. interestingly, the guarantees improve slightly over time, even though we allowed the adversary to compromise a fixed percentage of nodes, and thus, to compromise more nodes of its choice as the Tor network grows in size.

Anonymity guarantees depending on the ports. The ports requested by the user significantly impact the (recipient) anonymity guarantees. In Figure 14 we show the recipient anonymity guarantees depending on the number of compromised nodes for the 5'th of February. As settings we chose a multiplicative value of $\varepsilon = 0$ and we disabled guards and did not restrict the path selection to fast or stable nodes.

6.4 The impact of a multiplicative factor

The definition of ANOA introduces a multiplicative factor in addition to the normal additive factor (that often suffices to describe the success probability of an adversary). This factor allows for accounting for various events in which an adversary might gain information that may even lead to a non-negligible advantage without overestimating these events.

The experiments show that such a factor often only plays a minor role, as the probability to completely deanonymize a user is for most settings higher than the probability to just learn some information about them. Recipient anonymity in a setting with a weaker adversary, that compromises no, or only a very limited amount of nodes presents a noteworthy exception. Recall that for recipient anonymity we assume that the ISP of the user is compromised, which means that the adversary can see which entry node the user connects to. For different ports the probability of choosing these entry nodes, however, will be different, because they might also be possible exit nodes, or related to possible exit nodes. For PSTOR, an adversary that compromises no (only a very limited

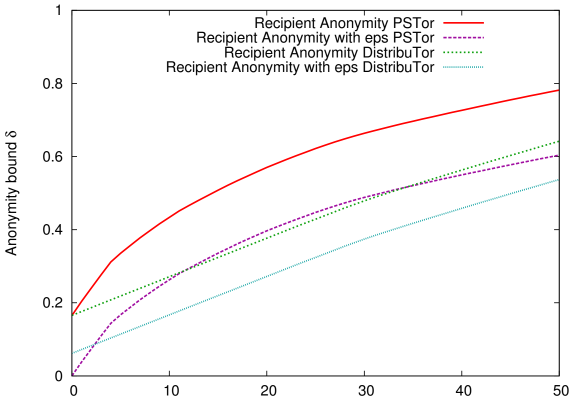

Figure 16: Impact of a multiplicative factor on recipient anonymity. We chose values for ε of $\varepsilon = 0.25$ for one setting and $\varepsilon = 0$ for the other setting. The graph depicts the value for δ computed by the monitor for both path selection algorithms. The scenarios are: HTTPS + IRC vs. HTTPS

number of) nodes can have already a non-negligible advantage in guessing which port a user might choose, which can either be expressed by a multiplicative factor and a δ of zero, or by a non-negligible δ. (See Figure 16.) Notice that for DISTRIBUTOR the value for δ does not reach zero, as the redistribution of exit bandwidth introduces a small (distinguishing) event in which the adversary can win without compromising nodes.

7 Conclusion & Future Work

This work presents a framework for rigorously assessing the degree of anonymity in the Tor network: MATOR. We carefully address the impact of user anonymity by real-life characteristics of Tor, such as its path selection algorithm, Tor consensus data, and the preferences and the connections of the user. The anonymity assessment is derived from a theoretical framework for anonymous communication networks. To obtain such a theoretical framework that suits our needs, we extended the ANOA framework [6]: a general framework for anonymous communication networks. Using MATOR together with Tor's publicly available consensus and server descriptor data, we developed the first real-time anonymity monitor. We apply this real-time monitor to archived data of the Tor network, using Tor Metrics data [29]. Based on the evaluation of these experiments, we propose an alternative path selection algorithm DISTRIBUTOR. We illustrate by our experiments that DISTRIBUTOR provides stronger anonymity guarantees without decreasing the overall performance of the Tor network.

A natural next step is the integration of MATOR to the actual Tor code. An interesting direction for future research is modeling the recently proposed congestion-aware path selection algorithm [32]. This path selection improves the overall performance of Tor, but reduces the anonymity guarantees of Tor. It would be great to see whether it is possible to compute in real-time a bound on the loss of anonymity loss of this more efficient path selection.

For future work, we also leave the application of the MATOR framework to Tor hidden services. A real-time anonymity monitor for a hidden service could be used to automatically disconnect from Tor whenever the anonymity bounds drop below a certain threshold.

Acknowledgments. We would like to thank the anonymous reviewers for their helpful comments and Marcin Slowik for revising the code and implementing MATOR for relationship anonymity. This work was supported by the German Ministry for Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA) and the German Universities Excellence Initiative.

References

[1] Source-code of MATor. available at `https://www.infsec.cs.uni-saarland.de/projects/anonymity-guarantees/mator.html`.

[2] SQLite. `http://www.sqlite.org/`.

[3] The Boost C+ Libraries. `http://www.boost.org`.

[4] Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha. LASTor: A Low-Latency AS-Aware Tor Client. In *Proc. of the 2012 IEEE Symposium on Security and Privacy (S& P)*, pages 476–490. IEEE Computer Society, 2012.

[5] Michael Backes, Ian Goldberg, Aniket Kate, and Esfandiar Mohammadi. Provably Secure and Practical Onion Routing. In *Proc. 26st IEEE Symposium on Computer Security Foundations (CSF)*, pages 369–385, 2012.

[6] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. Anoa: A framework for analyzing anonymous communication protocols. In *Computer Security Foundations Symposium (CSF), 2013 IEEE 26th*, pages 163–178. IEEE, 2013.

[7] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. AnoA: A Framework For Analyzing Anonymous Communication Protocols — Unified Definitions and Analyses of Anonymity Properties. IACR Cryptology ePrint Archive, Report 2014/087, 2014. available at `http://eprint.iacr.org/2014/087`.

[8] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching From a Distance: Website Fingerprinting Attacks and Defenses. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS)*, pages 605–616, 2012.

[9] Sambuddho Chakravarty, Angelos Stavrou, and Angelos D. Keromytis. Traffic Analysis against Low-Latency Anonymity Networks Using Available Bandwidth Estimation. In *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS)*, pages 249–267, 2010.

[10] Cynthia Dwork. Differential Privacy. In *ICALP (2)*, pages 1–12, 2006.

[11] Matthew Edman and Paul Syverson. As-awareness in tor path selection. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 380–389, New York, NY, USA, 2009. ACM.

[12] Nathan S. Evans, Roger Dingledine, and Christian Grothoff. A Practical Congestion Attack on Tor Using Long Paths. In *Proceedings of the 18th USENIX Security Symposium (USENIX)*, pages 33–50, 2009.

[13] J. Feigenbaum, A. Johnson, and P. F. Syverson. A Model of Onion Routing with Provable Anonymity. In *Proc. 11th Conference on Financial Cryptography and Data Security (FC)*, pages 57–71, 2007.

[14] J. Feigenbaum, A. Johnson, and P. F. Syverson. Probabilistic Analysis of Onion Routing in a Black-Box Model. In *Proc. 6th ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 1–10, 2007.

[15] Joan Feigenbaum, Aaron Johnson, and Paul F. Syverson. Probabilistic Analysis of Onion Routing in a Black-Box Model. *ACM Transactions on Information and System Security (TISSEC)*, 15(3):14, 2012.

[16] Nethanel Gelernter and Amir Herzberg. On the limits of provable anonymity. In *Proc. 12th ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 225–236, 2013.

[17] Yossi Gilad and Amir Herzberg. Spying in the Dark: TCP and Tor Traffic Analysis. In *Proceedings of the 12th Privacy Enhancing Technologies Symposiun (PETS)*, pages 100–119, 2012.

[18] Xun Gong, Negar Kiyavash, and Nikita Borisov. Fingerprinting Websites using Remote Traffic Analysis. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, pages 684–686, 2010.

[19] Alejandro Hevia and Daniele Micciancio. An Indistinguishability-Based Characterization of Anonymous Channels. In *Proc. 8th Privacy Enhancing Technologies Symposium (PETS)*, pages 24–43, 2008.

[20] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-Tin. How much anonymity does network latency leak? *ACM Transactions on Information and Systems Security (TISSEC)*, 13(2):13, 2010.

[21] Amir Houmansadr and Nikita Borisov. SWIRL: A Scalable Watermark to Detect Correlated Network Flows. In *Proceedings of the 18th Annual Network & Distributed System Security Symposium (NDSS)*, 2011.

[22] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users get routed: Traffic correlation on tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 337–348. ACM, 2013.

[23] Aaron M Johnson, Paul Syverson, Roger Dingledine, and Nick Mathewson. Trust-based anonymous communication: Adversary models and routing algorithms. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 175–186. ACM, 2011.

[24] Zhen Ling, Junzhou Luo, Yang Zhang, Ming Yang, Xinwen Fu, and Wei Yu. A Novel Network Delay based Side-Channel Attack: Modeling and Defense. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM)*, pages 2390–2398, 2012.

[25] Prateek Mittal, Ahmed Khurshid, Joshua Juen, Matthew Caesar, and Nikita Borisov. Stealthy Traffic Analysis of Low-Latency Anonymous Communication Using Throughput Fingerprinting. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, pages 215–226, 2011.

[26] Gavin O'Gorman and Stephen Blott. Improving Stream Correlation Attacks on Anonymous Networks. In *Proceedings of the 24th ACM Symposium on Applied Computing (SAC)*, pages 2024–2028, 2009.

[27] Lasse Øverlier and Paul Syverson. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, May 2006.

[28] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website Fingerprinting in Onion Routing based Anonymization Networks. In *Proceedings of the 10th ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 103–114, 2011.

[29] Tor Metrics Portal. `https://metrics.torproject.org/`. Accessed in May 2014.

[30] The Tor Project. `https://www.torproject.org/`. Accessed in May 2014.

[31] Chris Wacek, Henry Tan, Kevin S Bauer, and Micah Sherr. An empirical evaluation of relay selection in tor. In *Proc. 20th Annual Network & Distributed System Security Symposium (NDSS)*, 2013.

[32] Tao Wang, Kevin Bauer, Clara Forero, and Ian Goldberg. Congestion-aware path selection for tor. In *Financial Cryptography and Data Security*, pages 98–113. Springer, 2012.

[33] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective Attacks and Provable Defenses for Website Fingerprinting. In *Proc. 23th USENIX Security Symposium (USENIX)*, 2014.

[34] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. Defending Anonymous Communication Against Passive Logging Attacks. In *Proc. 24th IEEE Symposium on Security and Privacy*, pages 28–43, 2003.